UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Gerenciamento de Autoconfiguração em Redes com IPv6

Jussara Maria Bozzano

Florianópolis, setembro de 1998.

Gerenciamento de Autoconfiguração em Redes com IPv6

Jussara Maria Bozzano

Prof. Dr. Carlos Becker Westphall
Orientador

Área de Concentração: Gerência de Redes

Dissertação submetida à Universidade Federal de Santa Catarina para a obtenção do grau de Mestre em Ciência da Computação.

Florianópolis, setembro 1998.

Gerenciamento de Autoconfiguração em Redes com IPv6

Jussara Maria Bozzano

Esta dissertação foi julgada adequada para a obtenção do título de

MESTRE EM CIÊNCIA DA COMPUTAÇÃO

Especialidade **SISTEMAS DE COMPUTAÇÃO** e aprovada em sua forma final pelo Programa de **PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO**.

.....

Prof. Dr. Carlos Becker Westphall

Orientador

.....

Prof. Dr. Jorge Muniz Barreto

Coordenador do Curso de Pós-Graduação em Ciência da Computação

BANCA EXAMINADORA:

.....

Prof. Dr. Carlos Becker Westphall

Presidente

.....

Prof. Dr. Aloysio de Castro Pinto Pedroza

.....

Prof. Dr. João Bosco Mangueira Sobral

.....

Prof. Dr. Luiz Carlos Zancanella

.....

Prof. Dr. Roberto Willrich

AGRADECIMENTOS

A Deus.

A meus pais e irmãos.

A todos os meus amigos e colegas de trabalho.

A meu orientador e professor Carlos Becker Westphall.

ÍNDICE

LISTA DE ABREVIATURAS

AS - Autonomous System

ARP - Address Resolution Protocol

ARPA - Advanced Research Project Agency

ATM - Asynchronous Transfer Mode

BGP - Border Gateway Protocol

BOOTP - Bootstrap Protocol

BSD - Berkeley Software Distribuition

CATNIP - Common Archictecture for Internet

CIDR - Classless Inter-Domain Routing

CLNP - Connection-Less Network Protocol

CMIP - Common Management Information Protocol

DHCP - Dynamic Host Configuration Protocol

DHCPv6 - Dynamic Host Configuration Protocol version 6

DNS - Domain Name System

EGP - Exterior Gateway Protocol

EUI-48 - 48-Bit Global Identifier

EUI-64 - 64-Bit Global Identifier

FDDI - Fiber Distributed Data Interface

FTP - File Transfer Protocol

HDLC - High-level Data Link Control Protocol

HTTP - Hypertext Transfer Protocol

IAB - Internet Architecture Board

IANA - Internet Assigned Numbers Authority

ICMP - Internet Control Messages Protocol

ICMPv6 - Internet Control Messages Protocol version 6

IEEE - Institute of Electrical and Electronics Engineers

IETF - Internet Engineering Task Force

IGMP - Internet Group Management Protocol

ISO - International Organization for Standardization

IP	- Internet Protocol
IPv4	- Internet Protocol version 4
IPv6	- Internet Protocol version 6
IPAE	- IP Address Encapsulation
IPNG	- IP Next Generation
IPNGWG	- IP Next Generation Working Group
ITU-T	- Telecommunication Standardization Sector T of the International Telecommunication Union
MIB	- Management Information Base
MTU	- Media Transmission Unit
NAT	- Network Address Translator
NCP	- Network Control Protocol
NDP	- Neighbor Discovery Protocol
NIS	- Network Information Service
NLA	- Next-Level Aggregator
NSAP	- Network Service Address Point
NSF	- National Science Foundation
NSFNET	- National Science Foundation Net

NTP - Network Time Protocol

OSI - Open System Interconnection

OSPF - Open Shortest Path First

PDU - Protocol Data Unit

PIP - Paul's Internet Protocolo

PPP - Point-to-Point Protocolo

RARP - Reverse Address Resolution Protocol

RFC - Request for Comments

RIP - Route Information Protocol

RIPE - Réseaux IP Européens

SIP - Simple IP ou Steve's IP

SIPP - Simple IP Plus

SLA - Site-Level Aggregator

SLIP - Serial Line Interface Protocol

SLP - Service Location Protocol

SMTP - Simple Mail Transfer Protocol

SNMP - Simple Network Management Protocol

SNMPv1 - Simple Network Management Protocol version 1

SNMPv2 - Simple Network Management Protocol version 2

SNMPv3 - Simple Network Management Protocol version 3

TCP - Transmission Control Protocol

TCP/IP - Transmission Control Protocol/Internet Protocol

TFTP - Trivial File Transfer Protocol

TMN - Telecommunication Management Network

TLA - Top-Level Aggregator

TP/IX - Next Internet

TTL - Time to Live

TUBA - TCP and UDP over Bigger Address

UDP - User Datagram Protocol

WWW - World Wide Web

LISTA DE FIGURAS

RESUMO

A *Internet* tem crescido consideravelmente nos últimos anos. Além do crescimento, a complexidade de sua estrutura e de suas aplicações também aumentou. Como o projeto do IPv4 baseou-se em uma realidade compatível com a década de 70, as características deste protocolo não oferecem mais as condições necessárias para atender as necessidades da *Internet* impedindo que ela cresça e se desenvolva de modo estável [LAB97][HUI97].

Uma nova versão do protocolo IP foi adotada pela comunidade de engenheiros da *Internet*, o IPv6. Esta versão surgiu exatamente para suprir as limitações do IPv4, bem como introduzir e melhorar conceitos recentes como autoconfiguração, mobilidade, autenticação, entre outros.

Por se prever uma grande quantidade de dispositivos conectados à *Internet*, a o protocolo IP teve o tamanho de seu endereço aumentado de 32 bits para 128 bits. Essa mudança permitiu a implementação de mecanismos de autoconfiguração adequados, que desde já, mostram-se de grande importância e essenciais quando se imagina o tamanho das futuras redes.

A autoconfiguração traz agilidade necessária para que novos dispositivos sejam conectados à rede sem intervenção manual do administrador. Entretanto, a sua estrutura deve estar sob controle para que não se transforme em um problema. Mecanismos de controle e monitoração devem ser empregados para a manutenção de um adequado sistema de autoconfiguração.

Em redes IP os mecanismos de gerenciamento são, mais freqüentemente, implementados através do protocolo SNMP. Neste protocolo as informações de gerenciamento são armazenadas em estruturas ou bases de dados conhecidas como MIBs. Para gerenciar um determinado protocolo ou equipamento uma MIB específica deve ser gerada e nela serão definidos os elementos (objetos) deste protocolo que são passíveis de gerenciamento.

Este trabalho tem como objetivo apresentar um estudo detalhado dos métodos de autoconfiguração propostos para o protocolo IPv6, bem como propor a implementação de MIBs que atendem as necessidades genéricas de gerenciamento destes métodos e apresentar sua utilização.

Palavras Chaves

Internet, TCP/IP, IPv4, IPv6, ICMPv6, DNS, autoconfiguração *stateless*, autoconfiguração *stateful*, DHCPv6, SNMP.

ABSTRACT

The *Internet* has been growing considerably in the last years. Besides the growth, the complexity of its structure and of its applications it also increased. As the project of the IPv4, was based on a compatible reality with the decade of 70, the characteristics of this protocol don't offer more the necessary conditions to assist the needs of the *Internet* impeding that it grows and it is developed in a stable way.

A new version of the protocol IP was adopted by the engineers of the *Internet* community, the IPv6. This version appeared exactly to supply the limitations of the IPv4, as well as to introduce and to improve recent concepts as autoconfiguration, mobility, authentication, among others.

For foreseeing a great amount of devices connected to the *Internet*, the protocol IP had the size of its address increased from 32 bits to 128 bits. That change allowed the implementation of appropriate autoconfiguration mechanisms that at once are shown of great importance and essential when it imagines the size of the future nets.

The autoconfiguration brings necessary agility so that new devices are connected to the net without the administrator's manual intervention. However, its structure should be on control so that it doesn't become a problem. Control mechanisms and monitoration should be used for the maintenance of an adapting autoconfiguration system. In nets IP management mechanisms is mainly implemented through the protocol SNMP.

In this protocol the management information are stored in data bases called MIBs. For managing a certain protocol or equipment a specific MIB should be generated, through it will be defined the elements (objects) of this protocol that are manageable.

This work has as objective to present a detailed study of the autoconfiguration methods proposed for the protocol IPv6, as well as to propose the implementation of MIBs that assist the generic needs of management of these methods and to present its use.

Key Words

Internet, TCP/IP, IPv4, IPv6, ICMPv6, DNS, stateless autoconfiguration , stateful autoconfiguration , DHCPv6, SNMP.

1 INTRODUÇÃO

Por volta de 1978, apenas algumas universidades e centros de pesquisas estavam integrados à *Internet*. Seu objetivo principal era a troca de informações de pesquisas.

Dois acontecimentos fizeram com que esse quadro fosse alterado. Em 1991, o WWW (*World Wide Web*) foi desenvolvido por Tim Berners-Lee e 1993 surgiu o primeiro navegador *Web* (o *Mosaic*). Após o advento do WWW, o número de *hosts* na *Internet* aumentou de 313.000, ao final de 1990, para 2.056.000 ao final de 1993 [WIZ98]. Com esse fato, os usuários passaram a ter um novo perfil. A introdução da multimídia à *Internet* fez com que o acesso à rede se tornasse mais amigável ao público em geral. Grupos comerciais começaram a oferecer acesso à *Internet* para a comunidade. Indústrias, pequenas empresas e profissionais autônomos, passaram a utilizá-la como meio de divulgação de seus produtos.

Houve um crescimento explosivo no tamanho e na complexidade topológica da *Internet*. Este crescimento gerou uma série de problemas em sua infra-estrutura. Explosão e instabilidade nas tabelas de roteamento, redução considerável de endereços IPs disponíveis e sobrecarga de atividades de seus administradores. Além disso, as redes que compõe a *Internet* são alvos constantes de acessos indevidos, causados por sua frágil estrutura de segurança.

Vislumbrando um agravamento destes problemas, a IETF (*Internet Engineering Task Force*) resolveu ao final de 1993, convocar a comunidade de engenheiros da *Internet* para que estudassem e propusessem um novo protocolo IP (*Internet Protocol*). Foi formado o grupo *IPNGWG (Internet Protocol Next Generation Working Group)* para tratar unicamente de questões referente à nova geração do protocolo *Internet*.

Depois de várias pesquisas e debates polêmicos, foi adotado o IPv6 como o novo protocolo a ser utilizado na *Internet*. Este protocolo foi projetado para suplantar as dificuldades do protocolo anterior (IPv4), introduzindo e melhorando conceitos como autoconfiguração, segurança e mobilidade dos *hosts*.

Dentre as suas principais características, a autoconfiguração é uma das mais importantes, sendo considerada parte integral de qualquer implementação IPv6 [HUI97]. Os *hosts* não devem receber apenas uma configuração estática. O processo de autoconfiguração permite a mudança de endereço IP dinamicamente quando necessário, bem como a atribuição de vários endereços simultaneamente.

O mecanismo básico de autoconfiguração (autoconfiguração *stateless*) não necessita nenhuma intervenção do administrador, através dele o *host* recebe seu(s) endereço(s) e está apto para estabelecer conexões. Entretanto, a configuração de endereço não é o único parâmetro de configuração a ser considerado. Outros parâmetros como nome, servidor de nomes, e etc., são fundamentais para o acesso às aplicações disponíveis. Além disso, deve-se controlar o acesso à rede, que não pode ser indiscriminado. Este tipo de autoconfiguração (autoconfiguração *stateful* – DHCPv6) baseia-se no modelo cliente/servidor, sendo que nos servidores é que haverá a intervenção do administrador, definindo os parâmetros de configuração que devem ser repassados aos clientes, atendendo as suas necessidades, bem como adequando-os a política de configuração e segurança adotados.

A autoconfiguração traz agilidade necessária para que novos dispositivos sejam conectados à rede. Entretanto, a sua estrutura deve estar sob controle para que não se transforme em um problema, como por exemplo, inadequada configuração, sobrecarga de atividade, não disponibilização de serviços, entre outros. Mecanismos de controle e monitoração devem ser empregados para a manutenção de um adequando sistema de autoconfiguração. Através destes mecanismos pode-se obter informações, tratá-las, possibilitando um diagnóstico e um encaminhamento para as soluções dos problemas.

Em redes IP os mecanismos de gerenciamento são freqüentemente implementados através do protocolo SNMP. Neste protocolo as informações de gerenciamento são armazenadas em estruturas ou bases de dados conhecidas como MIBs. Para gerenciar um determinado protocolo ou equipamento, uma MIB específica deve ser gerada e nela serão definidos os elementos (objetos) deste protocolo que são passíveis de gerenciamento.

A construção de uma MIB depende do conhecimento detalhado das características e funções do elemento a ser gerenciado. Através de sua análise serão

definidas pelo menos as variáveis essenciais e suas inter-relações que permitirão um adequado gerenciamento deste elemento.

1.1 Organização do Trabalho

O trabalho está dividido como segue:

- No capítulo 2 será apresentada uma breve descrição da *Internet* e os conceitos básicos da arquitetura *Internet*;

- No capítulo 3 será apresentado o estudo da versão 4 do Protocolo IP (*Internet Protocol*), abordando suas principais funções.

- Será abordado no capítulo 4, o IP versão 6. Os novos conceitos serão introduzidos fazendo uma análise comparativa com o IPv4;

- Um estudo de Gerência de Redes será introduzido no capítulo 5, dando ênfase a gerência SNMP (*Simple Network Management Protocol*);

- No capítulo 6, será apresentado a tecnologia *Web* e Java, bem como, serão apresentadas as ferramentas de gerenciamento empregadas para o desenvolvimento deste projeto;

- Através do capítulo 7 são apresentados os conceitos e características detalhadas dos métodos de autoconfiguração IPv6;

- O capítulo 8 tem como objetivo descrever os módulos das MIBs desenvolvidas, apresentando a sua funcionalidade baseada nos protocolos de autoconfiguração;

- Conclusões, resultados, e perspectivas futuras são apresentados no capítulo 9;

- Implementações das MIBs podem ser observados através dos Anexos 1 e 2 .

2 *INTERNET*

A *Internet* é uma rede de computadores formada pela interconexão de diversas redes pertencentes a várias organizações, cuja amplitude é internacional, utilizando como protocolo de comunicação o TCP/IP (*Transmission Control Protocol/Internet Protocol*). Esta rede não é vinculada a nenhum país ou instituição.

O propósito da *Internet* é permitir que haja a interconexão das redes de computadores localizadas em qualquer parte do planeta, permitindo que todos os seus usuários possam efetuar trocas de informações através dela [HUI95].

Como citado anteriormente, a *Internet* não está vinculada a nenhuma instituição ou pessoa física. Entretanto, algumas entidades determinam as regras básicas para o seu funcionamento. A estas entidades, cabem definir e avaliar propostas de novos padrões para *Internet*, bem como administrar o espaço de endereçamento e a conectividade da mesma. As principais entidades envolvidas na regulamentação da *Internet* são: a IETF, a IANA (*Internet Assigned Numbers Authority)* e a IAB (*Internet Architecture Board).*

2.1 Histórico

A *Internet* teve início no final dos anos 60. Neste período, a Agência de Projetos de Pesquisa Avançada do Departamento de Defesa dos Estados Unidos (*ARPA-U.S. Department of Defense's Advanced Research Project Agency*) fundou uma rede de computadores. Esta rede foi denominada de ARPANET. Inicialmente estava limitada aos Estados Unidos, posteriormente, expandiu-se pelo mundo restringindo seu acesso a alguns centros de pesquisa. O objetivo da ARPANET era a colaboração, ou seja, os centros de pesquisas compartilhariam arquivos, programas e trocariam informações através do correio eletrônico. O protocolo de comunicação da ARPANET era o NCP (*Network Control Protocol*) e a velocidade do seu *link* era de 56Kbps.

Por volta de 1978, foi desenvolvido o protocolo de comunicação TCP/IP (*Transmition Control Protocol*) e, em 1980, passou a ser utilizado pela ARPANET, substituindo o NPC.

Em 1983, a pedido da ARPA, a Universidade de Berkeley adicionou em sua nova versão do Unix (4.2 BSD-*Berkeley Software Design*) o protocolo TCP/IP. O Unix BSD, era distribuído gratuitamente para as universidades. Deste modo, as universidades que receberam uma cópia do Unix tiveram possibilidade de interconectar-se à ARPANET.

A ARPANET cresceu, de poucos *host*s para milhares [COM92a]. Deste modo, esta rede passou a ser o *backbone* das subredes TCP/IP, e começou a ser denominada *Internet*, na sua globalidade.

No ano de 1988, o DARPA resolveu finalizar a ARPANET, pois considerou-a como um experimento e, como tal, havia alcançado os seus objetivos. Uma outra rede foi então fundada pela Fundação Nacional de Ciência dos Estados Unidos (*National Science Foundation - NSF*). Esta nova rede, chamada de NSFNET, substituiu a ARPANET , formando entao, o *backbone* da *Internet*.

Atualmente, a *Internet* interconecta milhões de *hosts* ao redor do mundo. A velocidade dos novos *backbones* comerciais suplanta em milhares de vezes a velocidade inicialmente apresentada pela extinta ARPANET.

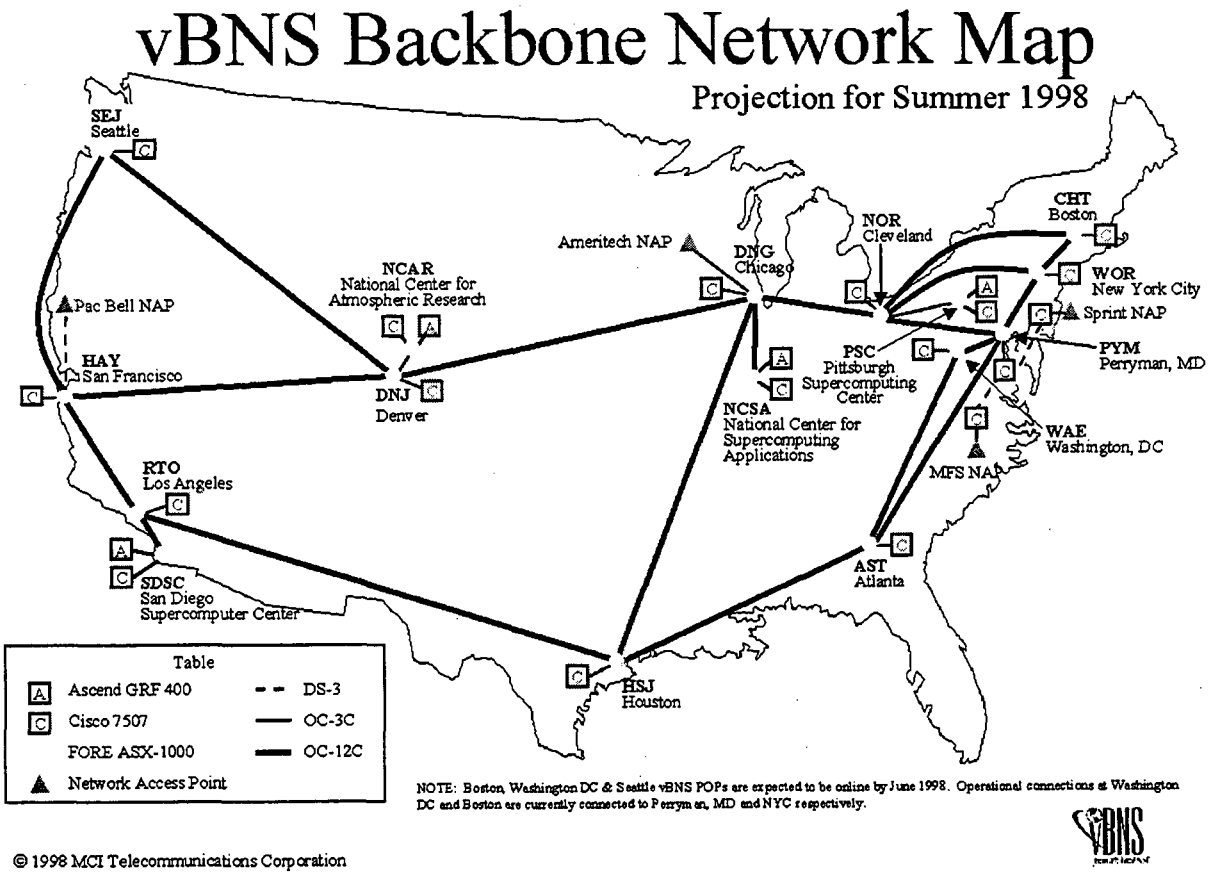
Com a popularização da *Internet*, grandes *backbones* comerciais foram se formando. Para refletir esta realidade, em 1995 a NSFNET deixou de ser o *backbone* principal da *Internet* passando a ser formado por múltiplos *backbones* comerciais. Mas a coordenação do *backbone Internet* está ainda sobre a responsabilidade da NFSNET.

2.2 Organização da *Internet*

A estrutura atual da *Internet* é baseada em pontos de acesso à rede (*Network Access Point - NAP*); estes NAP's estão conectados entre si formando a espinha

dorsal (*backbone*) da *Internet*. Através destes pontos de acesso, podem ser interconectados os provedores de serviço de rede (*Network Service Provider*), cuja função é realizar as conexões regionais.

A Figura 2.1 apresenta uma visão geral da arquitetura de rede da NFSNET. Ela é composta por três componentes primários: o vBNS (*very high speed Backbone Network Services*) que fornece conectividade em alta velocidade para colégios, universidades e instituições de pesquisa nos Estados Unidos; quatro NAPs e um RA (*Routing Arbiter*) que coordena o roteamento neste ambiente.



Fonte: http://www.merit.edu/nsf.architecture

Figura 2. 1 Mapa do *Backbone* vBNS fundado pela NFSNET

2.2 Arquitetura *Internet*

O protocolo TCP/IP é o protocolo que representa a arquitetura *Internet*. Este protocolo é constituído por uma estrutura mutável, que cresce e se desenvolve conforme as necessidades, por isso não apresenta uma estrutura bem definida e planejada. Deste modo, a arquitetura *Internet* apresenta uma estrutura flexível e adaptável a mudanças. Outra característica fundamental desta arquitetura é sua simplicidade. Por ser simples de implementar e facilmente maleável, a adesão do protocolo TCP/IP foi grande e o resultado é observado verificando-se a amplitude alcançada pela *Internet*. A arquitetura *Internet* é modelada em quatro camadas funcionais (ver Figura 2.2)[COM91]:

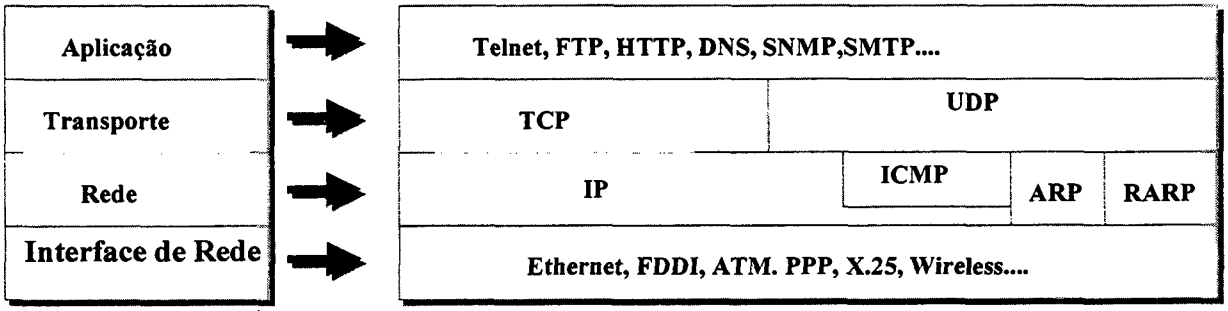


Figura 2. 2 Arquitetura *Internet*

- Interface de rede: também denominada de subrede de acesso. Esta camada representa as tecnologias de rede, como Ethernet, ATM (*Assyncronous Transfer Mode*), FDDI (*Fiber Distributed Data Interface)*, PPP (*Point to Point Protocol)*, etc. Praticamente todas as tecnologias de comunicação atuais pertencem a esta camada, pois o protocolo TCP/IP não define padrões fixos. A unidade básica de informação desta camada é o *frame* (quadro);

- Rede (IP): conhecida como camada *Internet*, ela permite que a *Internet* seja vista como uma grande rede virtual. Esta camada é a que determina a interconexão entre as redes da *Internet*. Encontramos nela 3 protocolos: IP (*Internet Protocol*), ICMP (*Internet Control Message Protocol)* e

ARP/RARP (*Address Resolution Protocol/Reverse Address Resolution).* O principal protocolo é o IP. Este protocolo não é orientado a conexão, ou seja, não estabelece um circuito virtual durante a comunicação. O IP não é um protocolo confiável, pois assume que essa confiabilidade seja garantida pela camada inferior (interface de rede e hardware) e superior (transporte). Não possui controle de fluxo e correção de erro. Estas funções devem ser fornecidas pelo protocolo TCP da camada de transporte. Se o protocolo de transporte for o UDP, estas funções devem ser garantidas pela camada de aplicação. A unidade básica de informação é chamada de datagrama. Uma visão completa desta camada é apresentada no próximo capítulo deste trabalho;

- Transporte (TCP): a camada de transporte permite a transferência de informações fim-a-fim. Por este motivo esta camada só será acessada pelos nós origem e destino. Os protocolos que a compõe são o UDP e o TCP. O protocolo UDP com relação a camada *Internet*, age apenas como uma interface. Este protocolo não é confiável, não possui controle de fluxo e não executa recuperação de erro; ele simplesmente multiplexa e demultiplexa datagramas IP permitindo a interação entre o IP e as aplicações. O UDP utiliza o conceito de portas[1] para direcionar os datagramas às aplicações específicas. O protocolo TCP é orientado a conexão, pois estabelece um circuito virtual durante a comunicação. O TCP é confiável, realiza controle de fluxo e executa recuperação de erros. Da mesma forma que o UDP, o TCP utiliza o conceito de portas para direcionar os datagramas às aplicações específicas. A unidade básica de informação da camada de transporte é chamada de segmento;

- Aplicação: é representada por processos (programas) que interagem diretamente com o usuário. Cada aplicação possui suas características próprias e são endereçadas através do número da porta que lhe é atribuída e

[1] Nº que identifica a aplicação. Por exemplo a aplicação telnet tem como valor padrão o número de porta 23.

do número IP do equipamento no qual está sendo executada. Como exemplo de aplicações podemos citar: Telnet, *FTP (File Transfer Protocol)*, SMTP (*Simple Mail Transfer Protocol)*, SNMP *(Simple Network Management Protocol)*, entre outras. A unidade básica de informação é chamada de mensagem.

3 PROTOCOLO *INTERNET* VERSÃO 4 (IPV4)

O objetivo do Protocolo *Internet* (*Internet Protocol* - IP) é fornecer as funções necessárias para que um datagrama seja entregue, a partir de um *host* origem, a um *host* destino, através de um sistema interconectado de redes. O IP não é orientado a conexão. É um protocolo não confiável, uma vez que os pacotes podem ser perdidos, duplicados e entregues fora da ordem sem que as camadas superiores sejam notificadas. Neste caso, assume-se que essa confiabilidade seja garantida pelas camadas imediatamente superiores e inferiores. Não possui controle de fluxo e correção de erro. Esta funções devem ser fornecidas pelo protocolo TCP da camada de transporte. Se o protocolo de transporte for o UDP, estas funções devem ser garantidas pela camada de aplicação. É um protocolo denominado *host*-a-*host*, pois é responsável pela entrega do datagrama para o próximo roteador ou *host* destino (ver Seção 3.1). Em função disto, o protocolo *Internet* implementa três funções básicas: endereçamento, roteamento e fragmentação de pacotes. Como o IP não é um protocolo orientado à conexão, ele trata cada datagrama como uma entidade independente, sem relação com qualquer outro datagrama. Existe controle de erro para dados, apenas em nível de cabeçalho. Deste modo, os erros são detectados e podem ser reportados através do ICMP (ver Seção3.5).

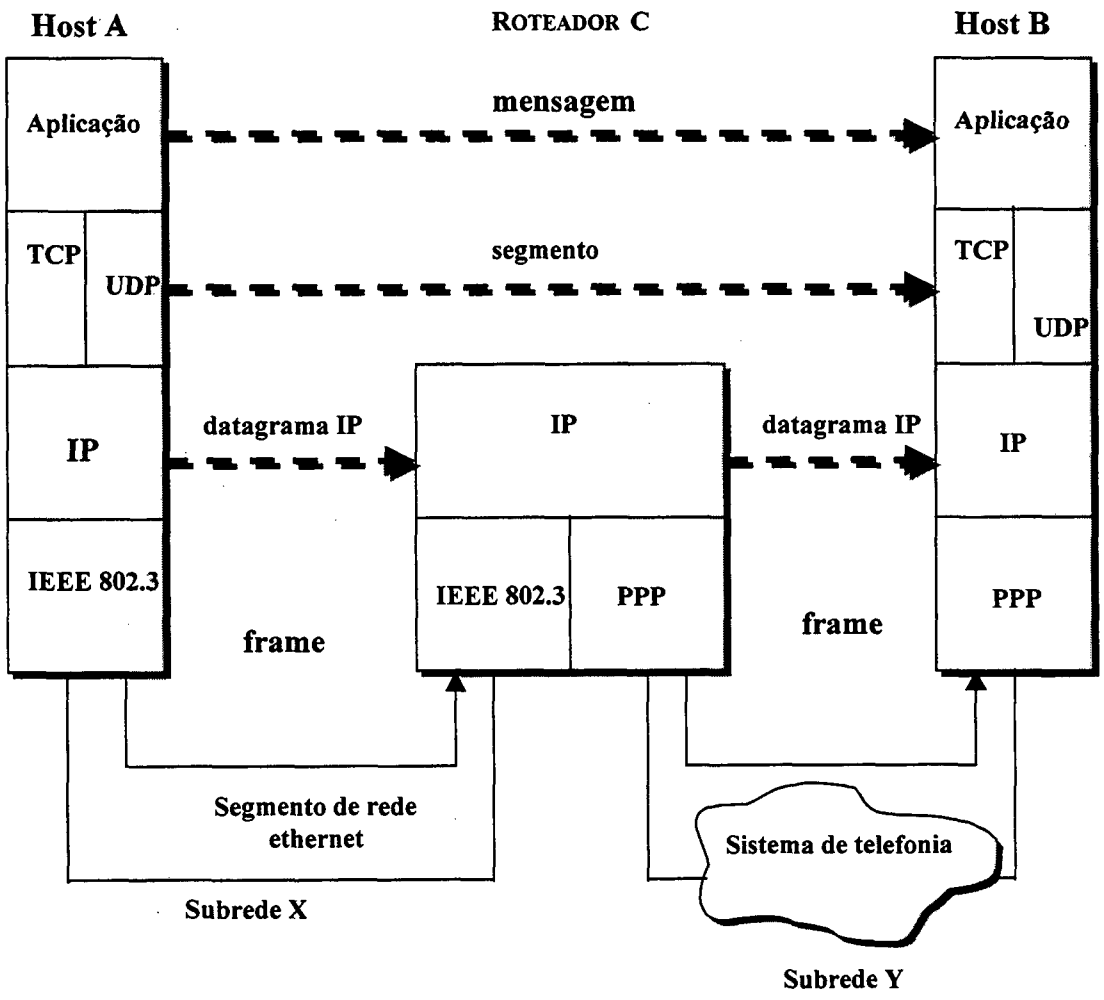


Figura 3. 1 Interconexão do Protocolo IP

3.1 Modelo de Operação

A *Internet* é composta por inúmeras redes, que são formadas por dois elementos básicos [IBM95]:

- *Host*: dispositivo que normalmente recebe e origina datagramas. Age como o ponto final (*end-point*) de uma comunicação. Tipicamente, possui uma única interface[1] de rede, com um único endereço IP associado;

- Roteador: é um *host* que interconecta duas ou mais redes a nível de camada de rede, e realiza o roteamento de pacotes entre elas. Para permitir a interconexão o roteador deve pertencer, no mínimo, a duas redes IP. A configuração padrão é possuir mais que uma interface de rede. Cada uma delas contendo um endereço IP, conectada a uma rede distinta.

Os roteadores terão que resolver a estrutura de roteamento associada aos protocolos de rede para saber para quem enviar os datagramas que chegam. Desta forma, devem determinar o próximo dispositivo (próximo *hop*[2]) para qual deverá encaminhar o datagrama.

Para definirmos o modelo de operação, devemos imaginar duas redes interconectadas. Uma aplicação acessa a rede através da camada de transporte, TCP ou UDP. A Figura 3.1 representa dois *hosts Internet*, conectados em diferentes redes, estas redes estão interconectadas através de um roteador C. Depois que a aplicação envia seus dados à camada de transporte, o programa desta camada encapsula em um ou vários pacotes, e os repassa para a camada de rede. A camada de rede encapsula o pacote recebido agregando um cabeçalho (ver Figura 3.2) onde é especificado o endereço origem, o endereço destino, e o próximo *hop* (neste caso o roteador C). O datagrama IP, então é repassado para interface de rede local. A interface de rede local cria um cabeçalho, e agrega a ele o datagrama IP.

[1] Representação lógica de *hardware* que permite o acesso ao meio físico. Por exemplo, placa de rede.

[2] *Host*/Roteador para onde deve ser enviado o datagrama.

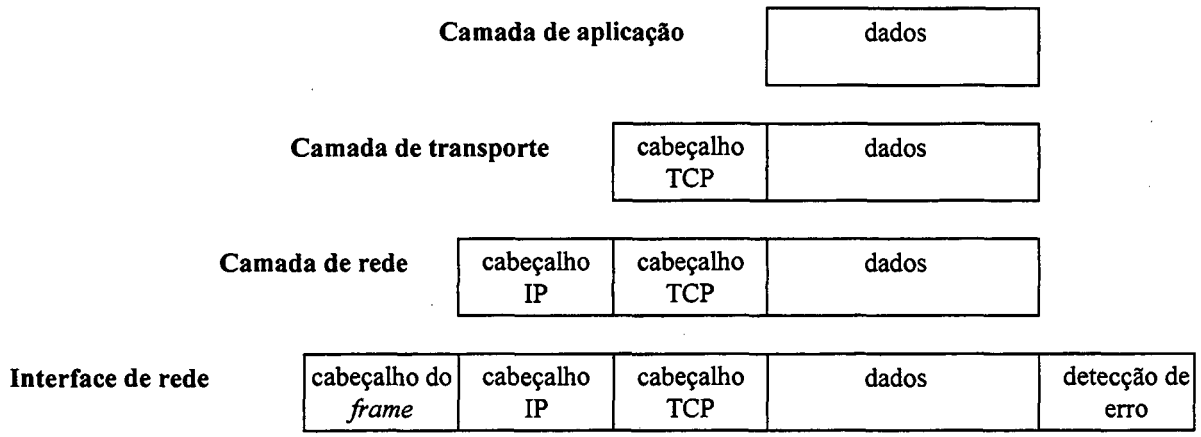


Figura 3. 2 Transmissão de Dados através das Camadas TCP/IP

O *frame* chega então ao roteador C através de sua interface de rede. O cabeçalho da interface da rede é removido e os dados, neste caso o datagrama é repassado para a camada *Internet*. A camada *Internet* verifica através do endereço IP que o datagrama não lhe é destinado, devendo ser repassado para um outro *host*, localizado na outra rede. A camada de rede determina, qual interface local deverá transmitir o datagrama. Esta interface cria um cabeçalho de rede, agrega o datagrama, e o envia ao *host* destino (*host* B). No *host* B, a camada da interface de rede remove o cabeçalho do *frame* e envia o datagrama para a camada de rede. A camada de rede avalia o cabeçalho deste datagrama e determina para qual protocolo de transporte (TCP ou UDP) devem ser enviado os dados. O protocolo da camada de transporte, por sua vez, remove o cabeçalho do pacote e verifica qual aplicação deverá receber a mensagem.

3.2 O datagrama IP

O datagrama IP é composto de um cabeçalho e um campo de dados. O cabeçalho do datagrama IP, tem um tamanho mínimo de 20 *bytes*. Isto porque, somente ao final dos 20 bytes têm-se os endereços origem e destino. O tamanho máximo do datagrama IP é de 64 Kbytes.

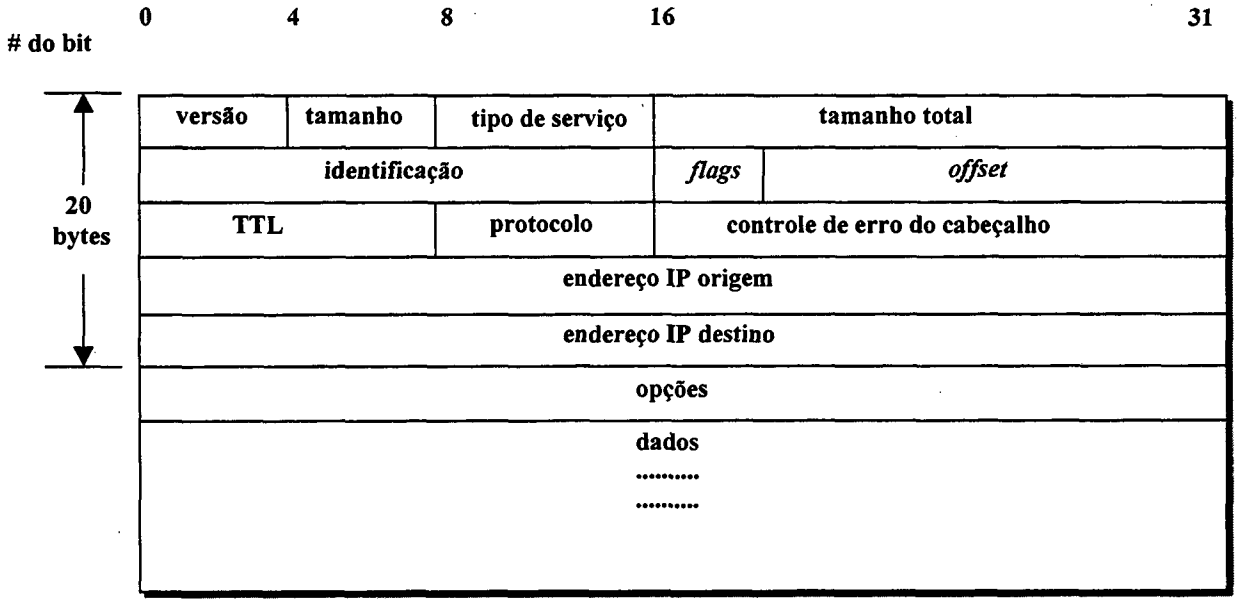


Figura 3. 3 Datagrama IP

A Figura 3.3 apresenta o formato do datagrama IP onde:

- versão: define o versão do protocolo IP (4 bits);

- tamanho: define o tamanho do cabeçalho IP, contados em quantidades de 32 bits;

- tipo de serviço: é uma indicação da qualidade de serviço requerida pelo datagrama IP. Este campo é subdividido em cinco subcampos através dos quais o emissor terá condições de definir a importância de cada datagrama, que tipo de transporte o datagrama necessita, ou seja, se necessita de um serviço sem demora, ou de um serviço com alto escoamento ou de um serviço de alta confiabilidade;

- tamanho total: define o tamanho total do datagrama, em bytes, do cabeçalho e dos dados.

- identificação, *flags* e *offset*: são campos utilizados pelo processo de fragmentação e montagem de datagramas;

- TTL (*Time to Live*): determina em segundos o tempo de vida que um datagrama deve permanecer na rede. A medida em segundos é de difícil controle, por isso, foi adotado que em cada roteador que o datagrama passar, este campo deverá ser decrementado em um. Eventualmente se este valor tornar-se zero o pacote é descartado. O valor inicial do TTL é definido pelo protocolo de mais alto nível, que cria o datagrama;

- protocolo: indica para qual protocolo de mais alto nível o datagrama deverá ser entregue;

- controle de erro do cabeçalho: esta checagem de erro abrange apenas o cabeçalho;

- endereço IP origem: endereço IP de 32 bits do *host* que envia o datagrama;

- endereço IP destino: endereço IP de 32 bits do *host* destino;

- opções: este campo apresenta tamanho variável. Objetiva permitir que alguns pacotes sejam transportados recebendo uma avaliação e um processamento distinto dos pacotes comuns. Um exemplo desta diferenciação é a requisição de um roteamento alternativo para uma determinada categoria de datagramas.

3.3 Endereçamento

Esta seção abordará uma das principais funções do protocolo IP que é o endereçamento. Cada nó na *Internet* necessita que um endereço IP lhe seja atribuído, pois só assim terá condições de se comunicar através de um sistema interconectado de redes.

3.3.1 Arquitetura de Endereçamento

Os *hosts* na *Internet* são identificados através de um endereço. Este endereço é conhecido como endereço IP, possui um tamanho de 32 bits e apresenta a seguinte notação: d.d.d.d. Onde d é a representação decimal de um valor composto por 8 bits. O endereço IP consiste de um par de números: **<número de rede><número do *host*>**.

O número de rede identifica a qual rede o *host* pertence. E o número do *host*, identifica o *host* na sua rede.

Existem 5 classes de endereços IP, que segmentam o espaço de endereçamento IP. As classes são mostradas na Figura 3.4.

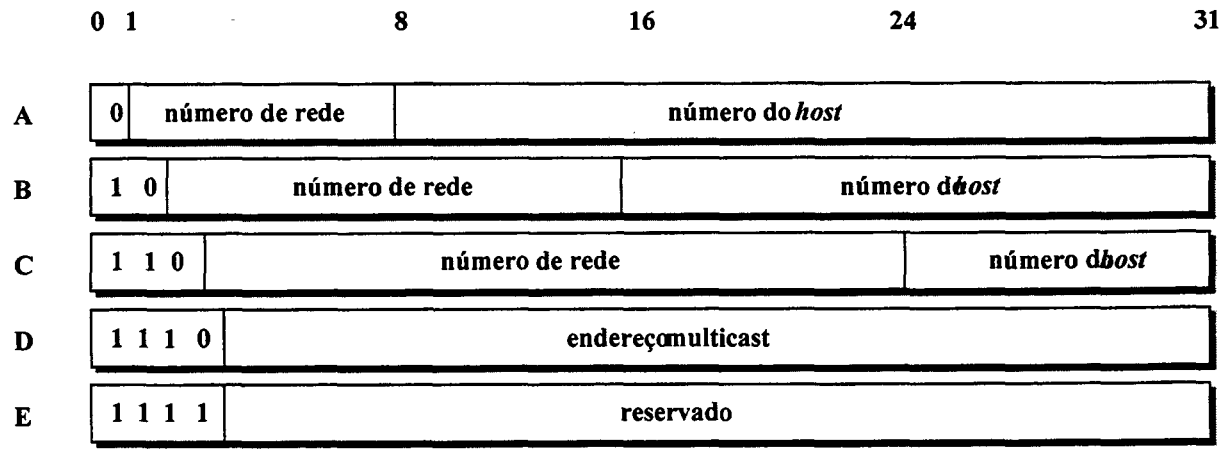


Figura 3. 4 Classes de Endereços IP

Os endereço de classe A usam 7 bits para identificar a rede e os 24 restantes identificam o *host*. Dessa forma, pode-se ter 126 redes de classe A com 16.777.214 *hosts*. Os endereços de classe B utilizam 14 bits para identificação da rede, resultando em um número máximo de 16.382 redes. Os 16 bits restantes, definem o número de 65.534 *hosts* possíveis. O campo número de rede dos endereços classe C é composto por 21 bits e o campo de *host* é composto por 8 bits. Possibilitando um total de 2.097.086 redes e 254 *hosts*. Os endereços de classe D estão reservados para *multicast* e os de classe E reservados para o futuro.

Existe um endereço IP especial chamado endereço *loopback*. Seu valor é 127.0.0.1 e é utilizado para que o *host* envie um pacote IP para si mesmo. Este endereço não poderá ser atribuído a uma interface.

Como exemplo de um endereço IP de um *host* temos 150.162.1.3 cujo valor binário é 10010110 10100010 00000001 00000011.

A entidade responsável pelo controle e distribuição dos endereços na *Internet* é o IANA. A uma organização é atribuído um endereço de rede exclusivo, e a cada *host* desta rede é atribuído um endereço IP que deve ser único em toda a *Internet*. Nenhum endereço IP pode ser atribuído a mais de uma interface. Por exemplo a redeUFSC recebeu o endereço de classe B **150.162** e a cada *host* da redeUFSC foi atribuído um endereço do tipo 150.162.1.3, 150.162.60.1, etc.

Como pôde ser observado acima, o endereços de classe A se ajustam às redes que possuem um grande número de *hosts* e os de classe C àquelas com um número reduzido. Entretanto, este tipo de hierarquia de dois níveis não foi suficiente para se ajustar a realidade da *Internet*. O que aconteceu é que o número de redes de tamanho pequeno e médio cresceram, devido principalmente à segmentação das redes locais em redes menores. Cada vez que uma instituição resolve criar uma nova rede, há a necessidade da requisição de uma nova classe de endereço IP. Isso ocorre mesmo que os números IPs de outras redes conectadas a esta mesma instituição não estejam esgotados. Para evitar este desperdício, criou-se o conceito de subrede. A solução foi sub-dividir a segunda parte do endereço IP, número de *host*, em dois subníveis: número de subrede e número de *host*. O endereço passa então a ser constituído por uma tríade de números: **<número de rede><número de subrede><número do *host*>**

Sempre que um endereço IP é atribuído a uma rede, cabe ao administrador desta rede utilizar o conceito de subrede para segmentá-la ou não.

Para definir o grau de segmentação de uma rede utiliza-se o conceito de máscara de rede. A máscara de rede utiliza a mesma notação que o número IP, e possui os mesmos 32 bits. Para determinar se um endereço pertence ou não a uma subrede utiliza-se a operação *comparison-under-mask*. Nesta operação os bits zero na máscara de subrede indicam a porção do número do *host*, bits 1 indicam a porção do número de rede.

Por exemplo, uma rede classe B que possui 16 bits no campo número de *host*, pode ser segmentada da seguinte forma: dos 16 bits reservados para *host*, 8 agora serão destinados ao campo de endereço de subrede. Para obtermos esta formatação a máscara deverá ser 255.255.255.0, binariamente 11111111.11111111.11111111.00000000

Um *host* com endereço IP 150.162.1.3 cuja máscara é igual a 255.255.255.0 podemos dizer que ele pertence a rede classe B 150.162 e a subrede 150.162.1.

Existem dois tipos de subrede:

- subrede estática: todas as subredes utilizam a mesma máscara de rede;
- subrede com tamanho variável: as subredes não terão necessariamente a mesma máscara de rede.

3.3.2 Resolução de Endereços

Antes que um datagrama IP possa ser enviado de um *host* origem para um *host* destino, ou repassado de um roteador para um *host* destino, este datagrama deve ser adicionado como campo de dados ao *frame* da camada física (ver Seção 3.1). O *frame* por sua vez conterá em seu cabeçalho um campo que indicará o endereço físico do *host* destino. Conclui-se, que para entregar um datagrama IP, além do endereço IP, o endereço físico do *host* destino deve ser conhecido. É fundamental salientar, que a importância do endereçamento IP reside na interconexão de redes. Não haveria necessidade de um endereço extra, como é o endereço IP, se não houvesse a interconexão

de redes. O sistema de endereçamento da camada física seria suficiente, pois haveria somente comunicação entre os *hosts* conectados a um mesmo *link.*

Como a *Internet* é formada por redes interconectadas, o sistema de endereçamento IP é fundamental, dessa forma, mecanismos de resolução de endereço IP em endereço físico devem ser empregados para que o datagrama seja entregue. Normalmente, cada tecnologia de rede possui seus próprios mecanismos de resolução de endereços, entretanto, este tema será abordado de modo genérico, introduzindo os métodos mais comumente empregados em redes locais e redes de longa distância ponto-a-ponto.

Redes Locais

O protocolo mais empregado em redes locais para descoberta automática de endereços físicos é protocolo ARP (*Address Resolution Protocol*) [PLU82]. O protocolo ARP originariamente foi projetado para redes locais que utilizavam a tecnologia Ethernet. Entretanto, atualmente, outras tecnologias tais como, Token-Ring e FDDI, também podem utilizá-lo.

Através do protocolo ARP, é feita a tradução ou mapeamento do endereço IP para seu endereço físico correspondente. O protocolo ARP utiliza um método baseado em *broadcast,* que é um sistema de entrega de pacotes onde há o envio do pacote a todos os dispositivos conectados a uma mesma rede.

Os *hosts* de um rede local mantém uma tabela que possui o mapeamento dos endereços IPs para endereços físicos, chamada de tabela ARP. A Figura 3.5 mostra o resultado da execução do comando "arp -a", que permite que o usuário verifique o conteúdo da tabela ARP.

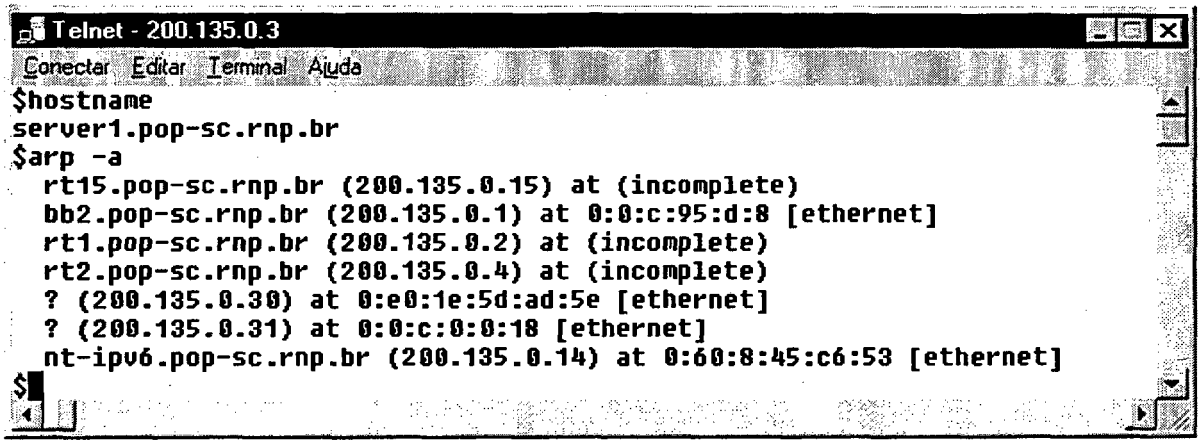


Figura 3. 5 Verificação do Conteúdo da Tabela ARP

Quando um *host* quiser inicializar a comunicação com um *host* vizinho (destino), ele procurará a entrada que possuir o endereço IP deste vizinho. Se o *host* origem possuir em sua tabela ARP o endereço físico do *host* destino, a mensagem é enviada diretamente utilizando o endereço físico disponível. Por outro lado, se não houver nenhum registro para aquele endereço IP na tabela ARP, o *host* envia um pedido ARP para todos os *hosts* vizinhos (*broadcast).* Este pedido ARP conterá como endereço IP destino, o endereço IP do *host* ao qual o *host* origem está tentando inicializar a comunicação. Todos os *hosts* conectados a rede vão receber o pedido ARP, entretanto, só o *host* que possuir o endereço IP igual ao endereço IP destino do pedido ARP é que responderá. Antes de enviar a resposta, o *host* destino atualizará a sua tabela ARP, com o endereço IP e respectivo endereço físico do *host* origem. Depois da atualização da tabela ARP, o *host* destino envia a resposta contendo os seus endereços físico e IP ao *host* requisitante (origem). Quando o *host* origem receber a resposta , ele atualizará a sua tabela ARP, tendo condições de transmitir os dados através da rede para o *host* destino.

As redes que utilizam a tecnologia ATM possuem uma máquina específica que mantém uma tabela similar a tabela ARP. Quando um *host* necessitar inicializar uma comunicação com um outro *host,* ele requisitará ao servidor o endereço físico do *host* destino.

Redes de Longa Distância Ponto-a-Ponto

Os protocolos mais comuns empregados em redes de longa distância ponto-a-ponto são: o HDLC (*High-level Data Link Control*), o PPP (*Point-to-Point*) e o SLIP (*Serial Line Interface Protocol*).

Os protocolos citados anteriormente são utilizados para que datagramas IP possam ser enviados através de uma ligação ponto-a-ponto entre um par de *hosts*, ou um par de roteadores, ou um *host* e um roteador. Por serem ponto-a-ponto, ou seja só existe um *host* vizinho, não utilizam o campo de endereço físico destino, não necessitando assim um mecanismo de resolução de endereço.

3.4 Fragmentação e Montagem de Datagramas

Um datagrama quando é transferido de um *host* a outro, normalmente é conduzido através de uma série de subredes. Estas subredes possivelmente não são similares podendo utilizar tecnologias diferentes. Cada tecnologia define sua MTU (*Maximum Transmission Unit*), ou seja, o tamanho máximo que um *frame* poderá ter. Estes valores são definidos segundo a taxa de dados máxima que poderá circular no meio e sua expectativa de erro. Dessa forma, não será qualquer tamanho de datagrama que circulará pelo meio. Para que o tamanho do datagrama se ajuste ao padrão de rede adotado, ele sofrerá fragmentação, dividindo-se em pacotes menores. Cada fragmento conterá uma cópia do cabeçalho do datagrama original com alguns dados modificados que serão utilizados no processo de montagem do datagrama. Cada um destes fragmentos será enviado como um datagrama normal. No *hosts* destino, os dados tem que ser remontados em um único datagrama. Cada um dos datagramas que chega ao *host* destino vai sendo armazenado em um *buffer*. Quando todos os fragmentos do datagrama chegarem, o datagrama original é restaurado e o processo continua, o cabeçalho é removido e os dados são enviados para a camada de transporte.

Esse processo de fragmentação e montagem de pacotes, tem se mostrado ineficiente nas modernas arquiteturas de redes [HUI95]. Modernas versões de TCP

têm implementado o *Path MTU Discovery*, que permite que o MTU suportado por todos os links seja dinamicamente determinado e, dessa forma, o nó origem envia apenas pacotes que não excedam o *Path MTU.*

3.5 ICMP (*Internet Control Message Protocol*)

Quando um roteador ou *host* destino necessitar informar ao *host* origem a respeito de algum erro no processamento do datagrama IP, ele utilizará o ICMP.

O ICMP, é visto pelo IP como um protocolo de mais alto nível. Entretanto, ele é parte integrante da camada de redes [POS81].

É necessário esclarecer que o ICMP não garante confiabilidade ao IP. Sua função é reportar alguns tipos de erros. Apesar disso os datagramas podem não chegar ao seu endereço destino sem que o *host* origem seja avisado.

As mensagens ICMP são enviadas no datagrama IP. O valor do campo protocolo do cabeçalho IP será 1. No campo de dados estarão as mensagens ICMP que podem reportar os seguintes eventos:

1. Destino Inalcançável

A mensagem ICMP de *host* inalcançável será enviada quando:

- rede/*host* inalcançável: este tipo de mensagem será enviada por um roteador. Ele conterá a informação, através de sua tabela de roteamento, que a rede é inalcançável, repassando esta informação ao *host* origem. Outra possibilidade é que a rede esteja alcançável, mas o *host* não. Se o roteador tiver esta informação poderá também repassá-la ao *host* origem;

- protocolo inalcançável: o *host* destino é alcançável, mas o protocolo IP é incapaz de repassar o datagrama para a camada superior devido a falhas na definição do protocolo superior ou porque uma porta destino não está ativa;

- fragmentação é necessária, entretanto o campo DF[3] está habilitado: neste caso o roteador descarta o datagrama e envia uma mensagem de destino inalcançável ao *host* origem.

2. Mensagem de Tempo Excedido

Este tipo de mensagem é enviada quando o roteador que está processando o datagrama verifica que o campo TTL possui um valor igual a zero, sendo então, necessário descatar este datagrama. Outra possibilidade de erro pode acontecer na montagem de datagramas fragmentados. Se o *host* destino não conseguir montar o datagrama dentro do tempo limite, o datagrama também é descartado.

3. Problemas com os Parâmetros

Se o *host* destino ou o roteador encontrou problemas no processamento do cabeçalho do datagrama, ele deverá ser descartado.

4. *Source Quench*

Este tipo de mensagem será enviada ao *host* origem por um roteador se o *buffer* onde ele enfileira os pacotes para saída na rede estiver esgotado. Da mesma forma o *host* destino enviará este tipo de mensagem ao *host* origem se as mensagens chegarem em uma velocidade muito grande que não permita seu processamento.

A mensagem *source quench* poderá ser enviada quando o *host* ou o roteador atingir seu limite no processamento de um datagrama.

5. Mensagem de Redirecionamento

Esta mensagem é enviada por um roteador r1 (ver Figura 3.6) quando ele conter a informação que o *host* origem (h1) e o próximo roteador estão na mesma rede. Desta forma o roteador r1 avisa o *host* origem (h1) para redirecionar as mensagens diretamente

[3] Subcampo do campo *flags* do cabeçalho IP que determina se o datagrama poderá sofrer fragmentação.

para o roteador r2. Possivelmente isto acontecerá por uma ausência no conteúdo da tabela de roteamento do *host* origem.

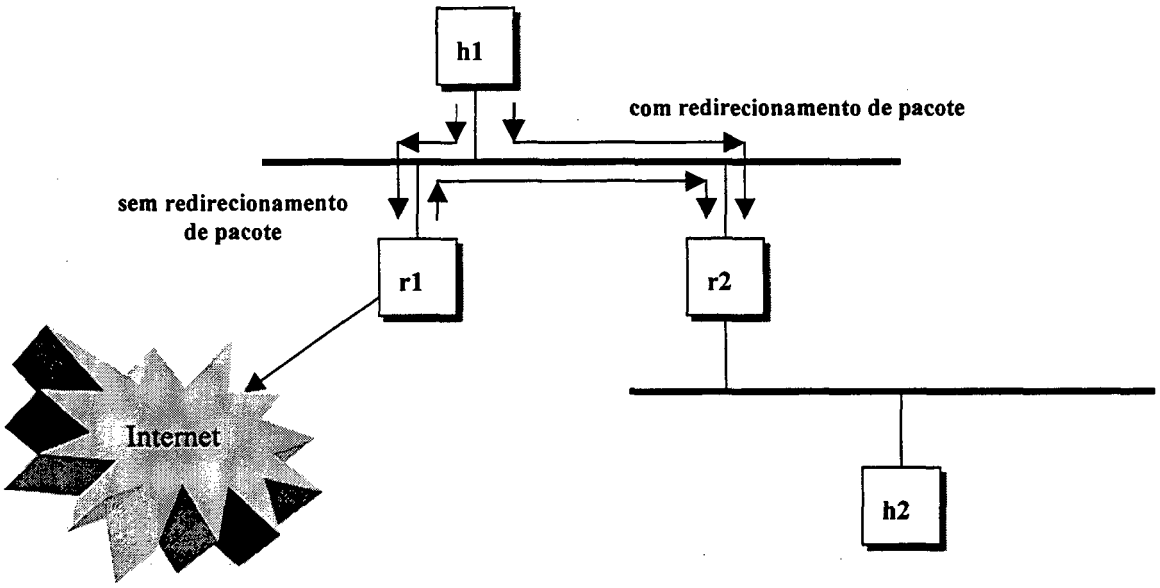


Figura 3. 6 Redirecionamento de Pacotes

6. Mensagem de *Echo* ou *Echo Reply*

Estes tipos de mensagens são utilizadas para o diagnóstico de redes, o ICMP determina se existe a comunicação entre dois *hosts*. O *host* origem utiliza o protocolo ICMP para enviar uma mensagem utilizando *echo*, e esta mensagem deve ser devolvida através de um *echo reply* pelo *host* destino. Um exemplo de aplicação que utiliza este recurso é o ping.

7. *TimeStamp* ou *TimeStamp Reply*

Esta mensagem é utilizada para medidas de desempenho e para depuração.

8. Aviso de Roteador e Solicitação de Roteador

- Aviso de Roteador: mensagem ICMP com a finalidade de permitir que os roteadores se identifiquem aos *hosts;*

- Solicitação: quando um *host* é inicializado e envia uma solicitação de descoberta de roteadores.

Estes 2 tipos de mensagens são usados se o *host* e o roteador suportarem o protocolo de descobrimento de roteador (*Router Discovery Protocol).*

3.6 Configuração com RARP, BOOTP e DHCP

Nesta seção serão apresentados os mecanismos mais empregados na configuração automatizada de nós *Internet.*

3.6.1 RARP (*Reverse Address Resolution Protocol*)

O protocolo RARP [FIN84] foi projetado para auxiliar um nó a encontrar seu endereço IP a partir do seu endereço físico. Este protocolo é baseado no modelo cliente/servidor. O servidor, conhecido como servidor RARP é um *host* que contém uma tabela com endereços físicos e seus respectivos endereços IPs associados, e o cliente geralmente é uma estação de trabalho *diskless*[4]. Uma *diskless* ao ser inicializada envia um pacote *broadcast* contendo seu endereço físico e requisitando um endereço IP. O servidor RARP presente na rede reconhece este pedido e verifica em sua tabela RARP se existe um endereço IP associado ao endereço físico do cliente especificado no pacote. Se existir um endereço IP associado, o servidor RARP responderá ao cliente.

Este protocolo tem sido substituído pelo protocolo BOOTP e sua versão mais completa o DHCP, por fornecerem mais parâmetros de configuração aos clientes.

[4] máquinas que operam sem a presença de uma disco rígido para armazenamento de dados.

3.6.2 BOOTP (*Bootstrap Protocol*)

O protocolo BOOTP [CRO85] além de possuir a funcionalidade do protocolo RARP, fornece a seus clientes outros parâmetros de configuração, especialmente informações de onde eles poderão obter o software para o *boot* (inicialização) do sistema.

O *host* cliente emite um pacote *broadcast* para o seu segmento de rede requisitando informações. O servidor BOOTP responde ao cliente, informando o seu endereço IP, e opcionalmente, a localização de um arquivo a ser obtido. O cliente por sua vez, utiliza o protocolo TFTP (*Trivial File Transfer Protocol*) para transferir o *software* para sua memória para posterior execução. Neste protocolo existe a necessidade que clientes e servidores estejam no mesmo segmento de rede.

3.6.3 DHCP (*Dynamic Host Configuration Protocol*)

O protocolo DHCP [DRO97] é uma evolução do protocolo BOOTP. Através deste protocolo, a configuração e a administração ficam simplificadas. No protocolo BOOTP, há a necessidade de definir parâmetros de configuração específicos para cada cliente. No caso do DHCP, os parâmetros de configuração podem ser definidos para um grupo de clientes. Por exemplo: o administrador pode simplesmente identificar um bloco de endereços IP que o servidor DHCP poderá alocar aos seus clientes. Além destas características, o DHCP incorpora outros parâmetros de configuração que podem ser repassados aos seus clientes que não são disponibilizados através do BOOTP.

O protocolo DHCP possui três modos de operação para a atribuição de endereços IP [FEI96]:

- Alocação Manual: um endereço IP que foi designado manualmente pelo administrador será permanentemente atribuído ao cliente;
- Alocação Automática: um endereço IP é selecionado de um grupo definido no servidor e será permanentemente atribuído ao cliente;

- Alocação Dinâmica: um endereço IP é atribuído a um cliente por um limitado período de tempo.

Neste protocolo não existe a necessidade que servidores e clientes pertençam ao mesmo segmento de rede.

3.7 Roteamento

Roteamento IP é uma importante função da camada de rede, pois é através desta função que existe a possibilidade de diferentes subredes se interconectarem.

Quando um datagrama alcança um *host* ou roteador, a primeira verificação a ser feita é se o datagrama está endereçado ao *host* local ou não. Se no cabeçalho do datagrama o campo do endereço IP destino contiver um endereço IP igual ao endereço IP do *host* local, o datagrama é enviado aos protocolos da camada superior. Entretanto, se o datagrama for endereçado a um outro *host* ele será tratado exatamente como um datagrama recém chegado da camada superior.

O datagrama será direcionado para o próximo *hop* de acordo com um algoritmo de verificação da tabela de roteamento. A tabela de roteamento (ver Figura 3.7) é uma lista de endereços alcançáveis direta ou indiretamente através da subrede. Cada uma das entradas desta tabela recebe o nome de rota. Em uma tabela de roteamento podemos encontrar:

- rotas diretas: para redes as quais o *host*/roteador está conectado;
- rotas indiretas: para as rede alcançáveis através de roteadores;
- rotas *default*: se não houver rota para uma determinada rede, nem rota direta ou indireta, o datagrama será enviado para o endereço do roteador para onde está apontando a rota *default*.

Rota	Próximo *hop*	Interface Lógica
default	150.162.1.232	en1
127	127.0.0.1	lo0
150.162.1	150.162.1.9	en1
150.162.9	150.162.9.1	150.162.9.1

Figura 3. 7 Exemplo de Tabela de Roteamento

O algoritmo de roteamento é esquematicamente apresentado na Figura 3.8.

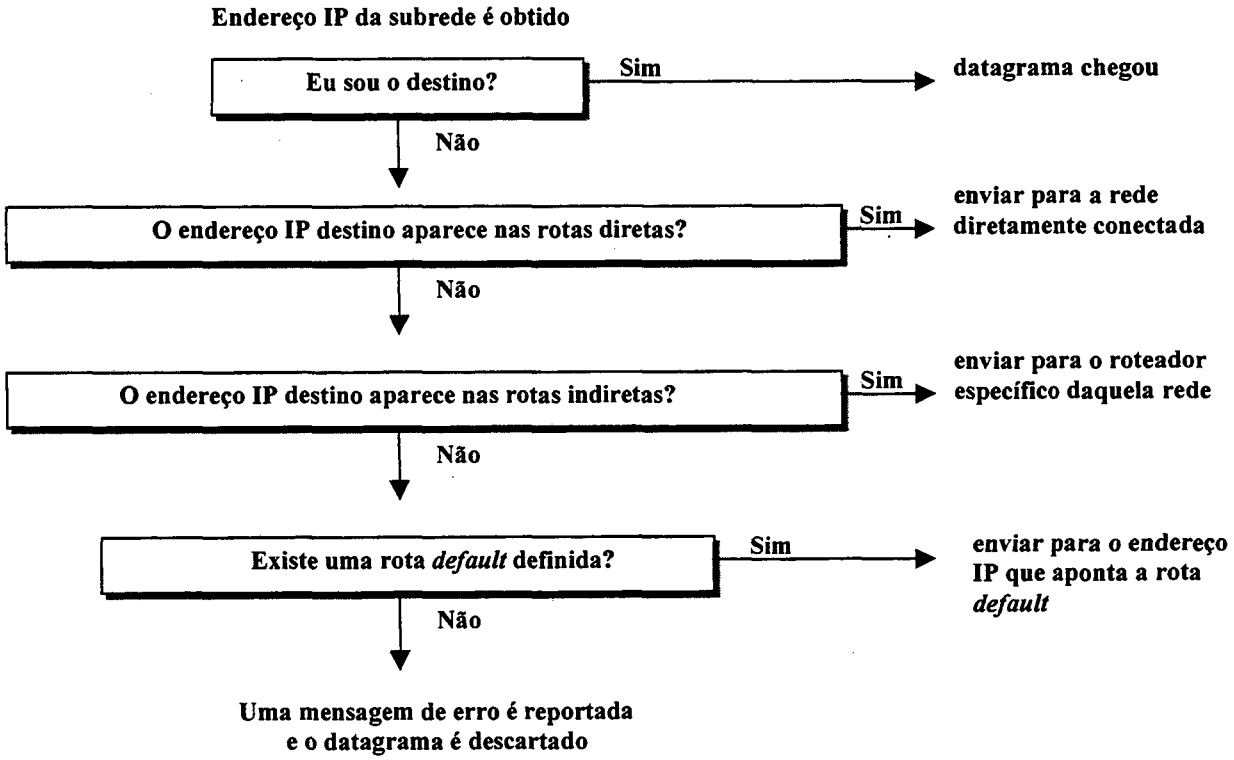


Figura 3. 8 Algoritmo de Roteamento

3.7.1 Protocolos de Roteamento

As tabelas de roteamento podem ser compostas estaticamente ou dinamicamente. Quando uma máquina é inicializada automaticamente, ela terá uma tabela mínima composta por rotas diretas.

Exemplo: Se host pertencente a subrede 200.135.14 e possuir endereço IP 200.135.14.2 então a sua tabela de rotas será similar a tabela de roteamento apresentada na Figura 3.9.

Rota	Próximo *hop*	Interface Lógica
127	127.0.0.1	lo0
200.135.14	200.135.14.2	en1

Figura 3. 9 Rotas Diretas

Entretanto, uma tabela composta unicamente por rotas diretas só permite a comunicação com as redes as quais o *host*/roteador está diretamente conectado.

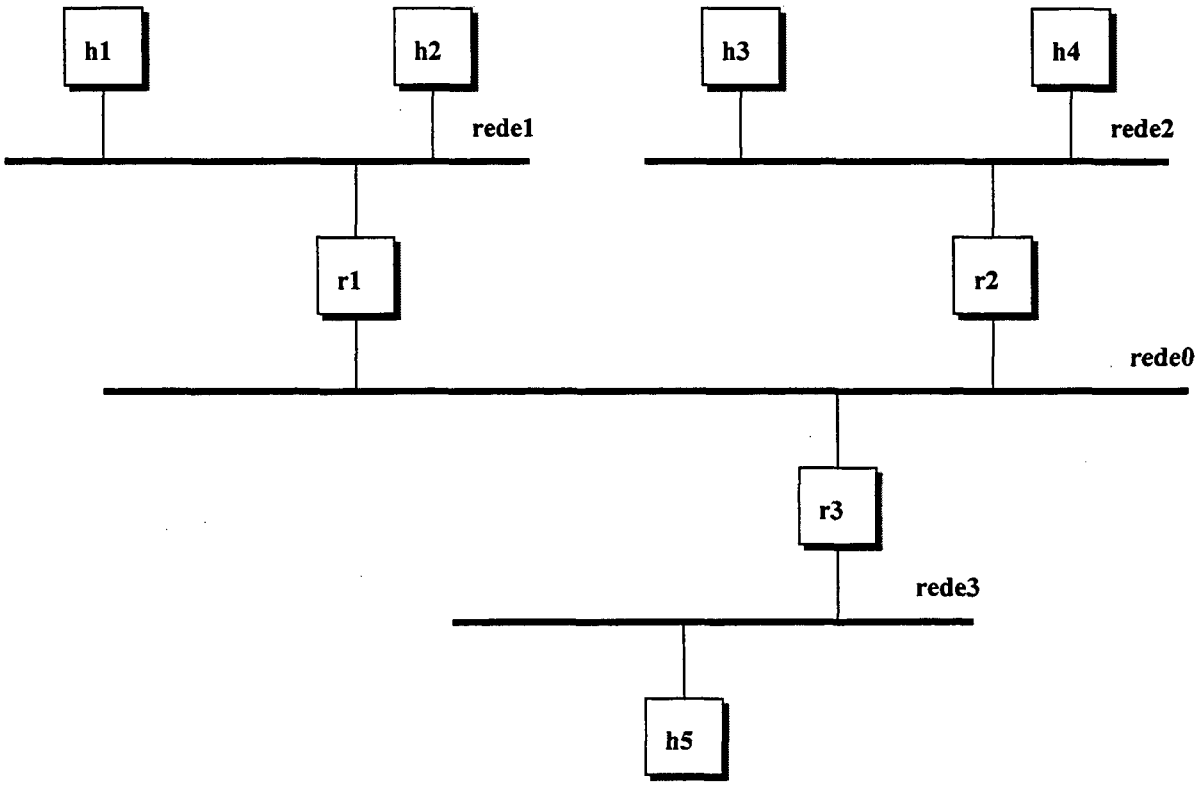


Figura 3. 10 Interconexão de Redes

Avaliando-se a Figura 3.10 e considerando que o *host* **h1** possui somente rotas diretas, se ele tiver que se comunicar com algum *host* de sua rede, não encontrará dificuldades. Entretanto, se ele tiver a necessidade de se comunicar com o *host* **h3**, a sua tabela de roteamento deverá conter uma rota que indique o caminho que o datagrama deverá tomar para atingir a outra subrede. Normalmente, em configurações simples como a do *host* **h1**, que possui um único roteador (**r1**) que faz a interconexão com outras subredes, a tabela de roteamento deverá conter apenas uma rota *default* apontando para **r1** além de suas rotas diretas. Esta tabela será suficiente para que seus datagramas sejam endereçados a qualquer ponto na *Internet*. Esta rota será inserida manualmente. Uma tabela simples como esta possivelmente não será suficiente para atender as necessidades de roteamento de **r1**. Ele está conectado a uma subrede composta por uma série de roteadores (**r1,r2,r3**), que por sua vez fazem interconexão com outras subredes. Somente uma rota *default,* não seria suficiente ou adequada para este tipo de topologia. Para que o roteador **r1** cumpra adequadamente suas funções ele deverá conter uma tabela de roteamento maior.

Este incremento pode ser feito de duas maneiras. A primeira delas seria adicionar uma rota estática para cada subrede, direcionando para o roteador que está conectado diretamente à ela. A segunda maneira, seria utilizar um protocolo de roteamento para obter estas rotas dinamicamente. Esta obtenção ocorre através da troca de informações entre os roteadores. Cada um deles notifica aos demais como está composta sua tabela de roteamento. Neste exemplo, o roteador **r1**, irá obter a rota para a rede **r2**, o mesmo acontecendo para as demais redes. A tabela de roteamento do roteador **r1** composta estática ou dinamicamente deverá conter basicamente as entradas apresentadas na Figura 3.11

Rota	Próximo *hop*
default	r3
rede2	r2
rede3	r3
rede1	r1

Figura 3. 11 Tabela de Roteamento

3.7.2 Protocolos de Roteamento Interior e Exterior

Com o crescimento da *Internet* foi necessário dividí-la em grupos de redes chamados de Sistemas Autônomos (*Autonomous System - AS*) com objetivo de diminuir a sobrecarga no roteamento e facilitar o gerenciamento da rede, computar rotas, distribuir novas versões de software e facilitar o isolamento de falhas.

Cada um destes sistemas autônomos é composto por um grupo de roteadores e redes que compartilham a mesma administração.

Os Sistemas Autônomos devem trocar tabelas de roteamento para que redes pertencentes a um AS possa se comunicar com redes de outro AS. Da mesma forma, interiormente a um AS, os roteadores devem trocar informações de roteamento. Os protocolos de roteamento entre AS são chamados de protocolos de roteamento exterior, e os que são processados interiormente, de protocolos de roteamento interior.

3.7.2.1 Protocolos de Roteamento Exterior

São protocolos de roteamento utilizados exclusivamente entre AS. Ao executá-los, os roteadores adquirem conhecimento a respeito das redes externas. Estas informações serão inseridas nas tabelas de roteamento local do AS.

O protocolo inicial utilizado pelo AS foi o EGP (*Exterior Gateway Protocol*). Seu projeto todo foi feito direcionado para as características da *Internet* da época com uma topologia estruturada em forma de árvore organizada ao redor de um *core*[5].

A *Internet* deixou de ser formada principalmente por instituições de pesquisa e educação. Ela começou a ser utilizada para fins comerciais. Redes comerciais se tornaram grandes *backbones* comerciais. A *Internet* passou a ser formada por esta mistura de *backbones* sem uma estrutura topológica bem definida. Foi desenvolvido então o BGP (*Border Gateway protocol*) para atender este novo tipo de topologia. O BGP vem sofrendo uma série de melhorias tentando se ajustar às necessidades da *Internet*. Atualmente está na versão 4, utilizando o método CIDR (ver Seção 3.7.3).

3.7.2.3 Protocolos de Roteamento Interior

São protocolos utilizados para troca de tabelas de roteamento entre roteadores de um mesmo AS. As tabelas de roteamento dentro de um AS devem incluir entradas cobrindo todos os possíveis destinos na *Internet*.

Os protocolos de roteamento mais comuns são o RIP (*Route Internet Protocol*) e o OSPF (*Open Shortest Path First*).

O RIP é ainda hoje o protocolo de roteamento mais utilizado no mundo. Apesar de ser um protocolo com eficiência limitada, pois causa uma sobrecarga na rede, suas tabelas de roteamento são trocadas a cada trinta segundos e não usa nenhuma técnica que permita agregação de rotas. É um protocolo cujo uso se justifica pela facilidade na sua configuração e implementação.

[5] Ponto central de acesso à *Internet*.

Por outro lado o OSPF, um protocolo mais recente, vem sendo cada vez mais adotado. As características positivas deste protocolo são:

- permite agregação de rotas ;
- após a primeira troca de tabelas só haverá nova troca se alguma alteração acontecer.

As dificuldades estão na implementação e configuração. Não são todos os sistemas operacionais que implementam.

3.7.3 Explosão nas Tabelas de Roteamento e Exaustão dos Endereços IP

Devido ao número limitado de endereços classe B, passou-se a designar pacotes de endereços da classe C para as organizações. Um endereço classe B permite que a rede tenha até 65.534 *hosts*. Um endereço classe C permite 254 hosts. Se a organização conta com uma estimativa de 3.000 *hosts*, repassar para ela um endereço classe B representaria um desperdício aproximado de 62.000 *hosts*. Para não haver este desperdício a solução seria distribuir 14 endereços classe C. Dessa forma, os roteadores teriam que ter 14 entradas na sua tabela de roteamento ao invés de uma. Esse aumento nas tabelas exige maior desempenho dos roteadores, sobrecarregando a rede e induzindo a possibilidade de erros mais facilmente. Note que este tipo de situação passou a ser comum na *Internet* gerando uma explosão nas tabelas de roteamento.

Uma solução para esse problema foi encontrado através do método de roteamento *CIDR (Classless Inter-Domain Routing)*.

O CIDR não faz o roteamento baseado na classe de endereço IP (por isso o termo *Classless* - sem classe), mas de acordo com os bits de mais alta ordem, chamado prefixo IP de 32 bits e uma máscara de rede de 32 bits, que juntos determinam o valor e o tamanho do prefixo IP: <endereço IP máscara de rede>. Por exemplo: <194.0.0.0 254.0.0.0>, representa que utiliza 7 bits cujo prefixo é 110001. Todas as redes que tiverem

este tipo de prefixo podem estar agrupadas. Por isso deve-se alocar grupos contínuos de endereço classe C para utilizar este recurso da agregação. Este processo é chamado de super-rede pois o roteamento é baseado nas máscaras de rede, que neste caso é menor que as máscaras de rede naturais, em oposição total a subredes cujas máscaras de rede são maiores que as naturais.

A agregação de endereços comuns é facilitada pelo escopo regional das autoridades de distribuição de endereços. Por exemplo, na Europa, o RIPE (*Réseaux IP Européens*) é o coordenador dos endereços IP e atribui os endereços de classe C segundo critérios de agregação de 194.0.0.0 até 195.255.255.255.

Se não se utilizasse o critério de agregação haveriam aproximadamente 380.000 entradas de rotas somente para referenciar as redes de classe C que inicializasse com 194 e 195.

Sem a utilização do CIDR provavelmente os endereços de classe B teriam se esgotado e o gerenciamento das tabelas de roteamento estaria seriamente comprometido.

3.7.4 NAT (*Network Address Translator*)

A introdução do conceito CIDR na definição da técnica de super redes conseguiu evitar o término prematuro dos endereços de classe B. Entretanto, o problema não está localizado unicamente no modelo inadequado da hierarquia de endereços *Internet*. A rede *Internet* vem crescendo dia-a-dia, além de hosts adicionais, novas subredes são implantadas, e a cada uma destas subredes no mínimo um endereço classe C será atribuído. O método NAT foi introduzido exatamente para ajudar a maximizar a utilização dos endereços IPs disponíveis.

Endereços Privados

A IANA reservou três seqüências de classes de endereços àquelas subredes que utilizam TCP/IP mas não tem acesso direto a *Internet*, conhecidas como redes

privadas. Este endereços não devem ser utilizados pelas subredes que acessam normalmente a *Internet*. Cabe aos administradores de subredes adicionarem filtros de acesso nos roteadores para estes endereços. Como não são roteáveis podem ser reutilizados. Por exemplo, a subrede de classe 192.168.1.0 (ver Figura 3.12) pode estar sendo utilizada por mais de uma empresa ou instituição.

Classe de Endereços	Sequência de Endereços de Rede
A	10.0.0.0 - 10.255.255.255
B	172.16.0.0 - 172.31.255.255
C	192.168.0.0 - 192.168.255.255

Figura 3. 12 Classes de Endereços Privados - Não Roteáveis

O método NAT traduz endereços privados não roteáveis em endereços registrados (públicos). Esta capacidade elimina a necessidade de atribuição de endereços IPs padrões para Intranets[6] que pretendem se conectarem à *Internet* e utilizam endereçamento privado. O NAT converte endereços privados em endereços registrados legalmente, com capacidade de enviar pacotes a *Internet* pública [MIL97].

Operação do NAT

O NAT operará no roteador limite entre a rede privada e a rede pública. A Figura 3.13 apresenta um modelo ilustrativo.

[6] coleção de redes interconectadas por roteadores e outros dispositivos que funcionam geralmente como uma única rede, pertencentes a uma empresa [MIL97], com acesso ou não à *Internet*

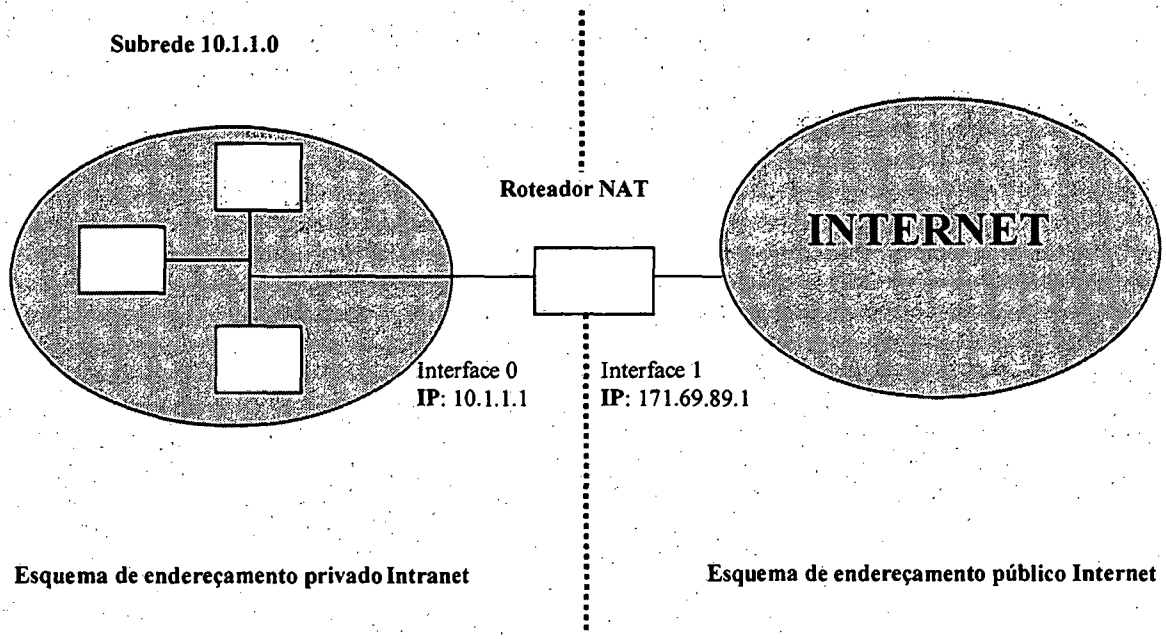


Figura 3. 13 Modelo NAT

Para demonstrar o funcionamento do NAT foi escolhido o método Endereço Local *Inside*. Não faz parte dos escopo deste trabalho o estudo das metodologias e configurações[7] NAT que podem ser adotadas. A Figura 3.14 ilustra o pacote IP transmitido da estação B para A e seu posterior retorno. Os pacotes são roteados da subrede 10.2.2.0, a qual pertence a estação B, para subrede 10.1.1.0 (**A1**) alcançando o roteador NAT com endereço 10.1.1.1 (**A2**). O pacote ao chegar ao roteador deve sofrer uma alteração no cabeçallho. O campo endereço IP origem que continha o valor 10.2.2.1, deve ser substituído por um IP público determinado estaticamente através de um tabela pré-definida ou através de uma tabela de alocação dinâmica (**A3,A4**). O endereço 10.2.2.1 estará associado a um endereço IP público a partir de um *pool* (conjunto) de endereços públicos alocados para este procedimento. Depois que o processo NAT foi realizado e o roteador determinou para qual interface o pacote deverá ser enviado (**A5**), a estação A recebe o pacote através do roteamento *Internet* (**A6**). A estação A sabe apenas que para retornar os pacotes à estação B ela deverá utilizar o endereço 171.69.89.1, especificado no

[7] NAT *of inside local address*, NAT *with* PAT *of global addressing*, NAT *handling of overlapping networks*, NAT *support for* TCP *load distribuition.*

campo endereço IP origem do pacote (mapeamento determinado ao 10.2.2.1 na tabela NAT). O roteamento *Internet* sabe que este pacote deverá ser enviado para o roteador NAT através de sua interface e0 (**B1, B2**). Depois que o pacote chegar ao roteador, o processo NAT checa a existência de um mapeamento prévio para o endereço 171.69.89.1 e determina que o endereço corresponde é o 10.2.2.1 (**B3**, **B4**), para onde o pacote deverá ser enviado através de sua interface e1 (**B5**), sendo roteado até a estação B (**B6**).

Apesar das características apresentadas acima que maximizam a utilização dos endereços IPs disponíveis, o NAT apresenta algumas desvantagens como:

- Dificuldade de traçar rotas, uma vez que o endereço dos pacotes pode mudar nas várias conexões;
- atraso e aumento na carga de processamento, por causa da tradução do endereço IP de cada pacote;
- força algumas aplicações que utilizam apenas o endereço IP (não utilizam o serviço de nomes) a pararem de funcionar, porque ele esconde os endereços IP fim-a-fim.

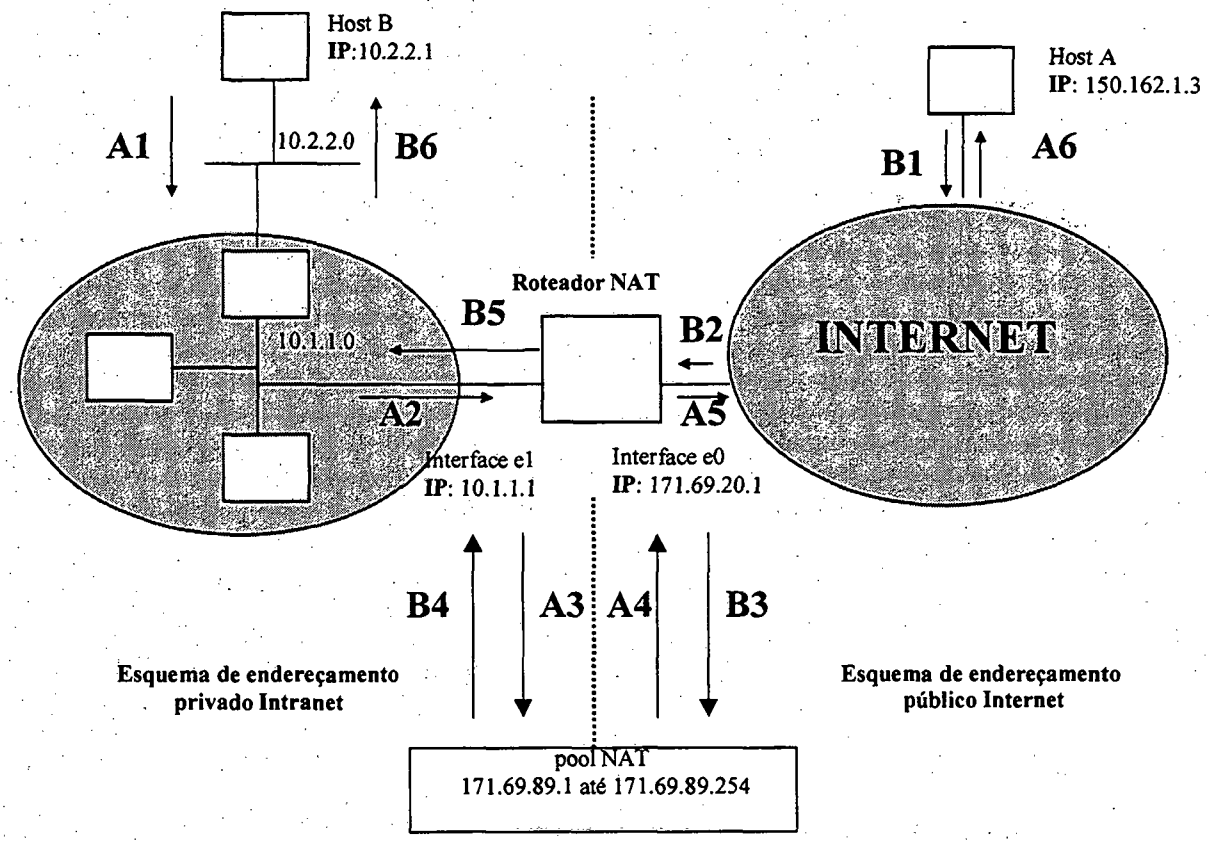


Figura 3. 14 NAT de Endereço Local

3.8 Serviço de Domínio de Nomes (*Domain Name Service – DNS*)

Cada *host* na *Internet* é conhecido através de seu endereço IP. Portanto para acessarmos um *host* devemos conhecer seus respectivos endereços IPs. Números nem sempre são fáceis de memorizar. Partindo deste princípio, foi decidido que cada endereço IP na *Internet* teria seu IP mapeado para um nome. As máquinas passaram a ter endereço e nome IP. No início a rede *Internet* era pequena, composta por uma centena de *hosts*. O mapeamento ficava localizado em tabelas de *hosts*. Cada *host* tinha sua tabela (ver Figura 3.15). Com o crescimento da *Internet*, o número de *hosts* aumentou. Em agosto de 1981 a tabela de *hosts* contava com 213 entradas, em agosto 1983 já eram 562 [WIZ98]. Esse tipo de sistema manual, começou a ficar inviável. Então, em 1984 foi desenvolvido e adotado o DNS. Esse sistema é uma base de dados distribuída que possui uma estrutura

hierárquica (ver Figura 3.16). Esta estrutura permite o controle local dos segmentos que compõem a base de dados. A estrutura do DNS é similar a uma árvore invertida. Cada um dos nós da árvore representa uma parte de toda a base de dados, chamada de domínio. Cada domínio recebe um nome e pode ser dividido em outros subdomínios. Os nomes dos domínios representam sua posição na base de dados e normalmente são administrados por diferentes organizações.

Cada *host* ou rede tem um nome de domínio, que fornece informações sobre o *host*. Tais informações são variadas, e compõem uma base de registros (nome, funções desempenhadas) .

Endereço IP	Nome
200.143.13.2	marte
125.162.18.4	jupiter
142.176.11.5	venus

Figura 3. 15 Tabela de *Hosts*

Os nomes dos domínios são sempre lidos do nó em direção a raiz. Observando a Figura 3.16, o nome do domínio será lido ufsc.br. e não .br.ufsc.

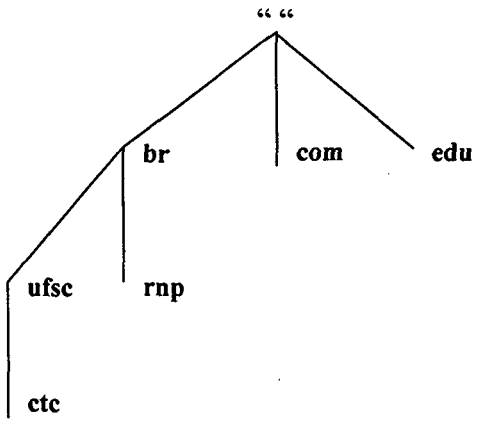


Figura 3. 16 Estrutura de Domínios do DNS

Todos os dados que compõe os Serviço de Nomes estão armazenados em arquivos tabelas. Como as informações não cumprem o papel único de

traduzir IP em nome e vice-versa, definiu-se o conceito de *resource records*. Cada um desses *resource records*, definirá o objetivo da informação.

O principal deles é o A (*address*), que faz o mapeamento nome em endereço IP:

Exemplo: npd.ufsc.br. A 150.162.1.3

Existe também o mapeamento reverso cujo *resource record* é o PTR (*pointer*) que traduz um número IP para um nome.

Exemplo: 3.1.162.150.in-addr.arpa. PTR npd.ufsc.br.

3.9 Segurança

Segurança em uma rede de computadores é a garantia de proteção dos dados e equipamentos contra as ações de usuários não autorizados, erros de *software*, vírus de computadores, falhas no *hardware* e todos os outros aspectos que comprometam ou prejudiquem o bom funcionamento do sistema e a confidencialidade dos dados.

Inicialmente as redes de computadores foram desenvolvidas para permitir o compartilhamento de informações e de equipamentos em universidades e instituições governamentais sem muita preocupação com a segurança. Porém, atualmente as redes são utilizadas por muitas pessoas e permitem executar operações de risco como operações bancárias, compras com cartão de crédito, transferência de informações confidenciais a respeito de assuntos militares, governamentais e de investigação policial, tornando-se imprescindível garantir a segurança das redes.

As atividades dos invasores são muitas, motivo pelo qual torna-se necessário garantir a segurança em todos os aspectos. As atividades mais comuns são :

- Leitura, remoção ou alteração das mensagens de correio eletrônico;
- Alterar ou remover aspectos de configuração, ou roubar dados;
- Descobrir as estratégias militares, policiais ou de empresas;

- Desfalcar dinheiro de empresas;
- Provocar algum dano ou prejuízo a alguma pessoa ou empresa por vingança;
- Roubar números de cartão de crédito.

3.9.1 Áreas de Segurança

A segurança de redes e sistemas de computação divide-se em áreas que se relacionam entre si:

- confidencialidade: garante que usuários não autorizados não tenham acesso às informações;
- autenticação: identifica o usuário caso ele seja cadastrado e nega acesso caso contrário;
- não repudiação: trata de assinaturas, ou seja : garante que a informação enviada é legítima e não falsa;
- controle de integridade: garante que a informação é verdadeira e que não foi alterada;
- consistência: assegura que o sistema irá funcionar de modo correto. Um erro de *software*, por exemplo, pode mudar o comportamento do sistema, causando danos;
- disponibilidade: garante que o sistema esteja disponível para seus usuários;
- controle: permite controlar os acessos no sistema, e em casos de acesso indevido, ratificar as falhas de segurança que viabilizaram o acesso não autorizado;
- auditoria: viabiliza meios de monitorar os acessos, indicando através de *logs*, o usuário que fez o acesso, as ações que realizou, a data e hora, etc.

3.9.2 Tipos de Ataques

Um sistema ou uma rede pode sofrer diversos tipo de ataques, sendo que os mais comuns são:

- modificação: alteração do conteúdo da mensagem;
- personificação: uma entidade se faz passar por outra para obter privilégios;
- *replay* : a mensagem é interceptada e, após certo tempo é transmitida;
- impedimento de serviço: ocorre se determinada entidade não executa sua função adequadamente, ou impede outras entidades de executarem suas funções;
- ataques internos: ocorre se o usuário do próprio sistema utiliza seus privilégios para executar ações indevidas ou não autorizadas;
- cavalos de tróia : ocorre quando um processo executa, além de suas funções reais, ações não autorizadas;
- armadilhas (*trapdoor*): um processo é alterado de forma a produzir efeitos não autorizados em resposta a um comando emitido pelo invasor.

3.9.3 Políticas de Segurança

A Política de Segurança define as prioridades na segurança do sistema e as regras que devem ser seguidas para garantir que o sistema seja confiável. Os padrões são definidos para auxiliar as políticas de segurança.

3.9.4 Criptografia

A criptografia é um mecanismo de segurança que modifica a mensagem a ser enviada, de forma que ela não possa ser facilmente identificada. Ao chegar ao destino, a mensagem é decriptografada, ou seja, volta ao seu formato normal, através da utilização de uma chave.

A mensagem é modificada por uma função, que utiliza uma chave como parâmetro. Assim, aplicando-se o mesmo método de criptografia em determinado texto, obtém-se textos cifrados distintos se as chaves utilizadas como parâmetro forem diferentes.

3.9.5. Aspectos de Segurança no Protocolo *Internet*

O IP foi desenvolvido em uma época em que não haviam tantos problemas de segurança como atualmente. Seu principal objetivo é o de transmitir pacotes de um computador a outro.

O protocolo IP foi preparado para solucionar falhas de hardware e software, problemas nas transmissão, ou seja , aspectos relacionados com seus objetivos. Por outro lado, não garante segurança contra ataques propositais, pois seus desenvolvedores não acreditavam que os próprios usuários teriam intenções de causar danos ao sistema.

O principal problema de segurança no IP é que não há autenticação, assim, não é possível saber com certeza se o *host* de origem é realmente o *host* que enviou o pacote.

3.9.5.1 Falhas na Segurança do IP

3.9.5.1.1 *NETWORK SNIFFERS*

A informação que trafega pela rede pode ser interceptada por *network sniffers*, que são programas que capturam e gravam os n primeiros caracteres dos pacotes transmitidos pela rede. Os *sniffers* são geralmente utilizados para capturar senhas.

3.9.5.1.2 IP *SPOOFING*

É um tipo de ataque em que o invasor utiliza um endereço de origem falso com o intuito de obter privilégios e executar ações indevidas sem ser rastreado.

3.9.5.1.3 *CONNECTING HIJACKING*

Através deste tipo de ataque um *hosts* pode se infiltrar em uma conexão entre dois *hosts*. Este ataque também é uma variação do IP *spoofing*. O *host* invasor se aproveita do estado não sincronizado entre os dois *hosts* (estado em que os *hosts* não concordam com as seqüências de números TCP) e envia a seqüência de números esperada pelo *hosts* receptor, interferindo na conexão.

A partir do momento que o *host* invasor se infiltra na conexão, o *host* de origem ignora as mensagens provenientes do *host* de destino e vice-versa , pois o *host* receptor agora utiliza a seqüência de números do *host* invasor e espera esta mesma seqüência do *host* de origem.

3.9.5.1.4 ATAQUE ATRAVÉS DE DNS

Nos ataques a segurança do DNS o invasor pode utilizar-se da técnica de tradução de números IP em nomes e forjar uma falsa configuração de modo que o seu próprio IP corresponda a determinado domínio, onde o invasor pretende se infiltrar.

Os servidores geralmente permitem acesso para as máquinas que possuem o mesmo domínio.

3.9.5.1.5 Ataques através do Protocolo de Roteamento RIP

Como no protocolo RIP não há autenticação, ou outra forma de garantir segurança, o invasor pode forjar um pacote RIP e especificar o próprio *host* como a rota de caminho mais curto, para que os pacotes passem pelo seu *host*. Além disso, um *host* pode se fazer passar por outro, de forma que todos as mensagens destinadas ao *host* legítimo sejam roteadas para o *host* invasor.

3.9.6 Soluções Propostas para os Problemas de Segurança

A versão 4 do protocolo IP apresenta falhas na segurança, porém, existem meios de evitar alguns ataques.

Atualmente estão disponíveis no mercado vários *software* que impedem ou dificultam os ataques conhecidos. Alguns deles podem até ser adquiridos gratuitamente através da *Internet*. No momento em que surge um novo modo de afetar a segurança, os pesquisadores procuram um meio de sanar o problema, ou pelo menos reduzir as chances de sucesso do ataque.

A ausência de autenticação no IP é o principal motivo de muitos problemas relacionados à segurança. É possível diminuir as possibilidades de ataque restringindo o número de *host* permitidos a acessar as máquinas onde estão os serviços mais importantes.

Outro modo de proteger o sistema é instalando TCP *wrappers*, programas que são executados antes que qualquer serviço seja inicializado. O TCP *wrappers* faz uma série de verificações de segurança e registra o *host* que requisitou o serviço e qual foi o serviço requisitado. Depois de tomar estas medidas de segurança o TCP *wrapper* inicializa o serviço solicitado normalmente.

A garantia de que os pacotes recebidos não são prejudiciais a segurança da rede, pode ser maximizada através da utilização de *firewalls*, que são programas que filtram os pacotes, ou seja , os analisam , aceitando pacotes confiáveis e descartando pacotes suspeitos.

É possível proteger o sistema adicionando autenticação associada a criptografia na camada de aplicação. O Sistema *Kerberos (Kerberos Autentication System)* utiliza algoritmos que garantem a legitimidade do *host* e criptografam as mensagens transmitidas.

A informação também pode ser preservada através da criptografia de todos os pacotes IP na camada de rede . Uma técnica utilizada para este tipo de criptografia é o SKIP (*Simple Key Management for Internet Protocol*).

4 PROTOCOLO *INTERNET* VERSÃO 6 (IPV6)

A *Internet* está crescendo em ritmo exponencial. O número de *hosts* conectados a ela é duplicado a cada ano e tudo indica que esta taxa de crescimento deve permanecer até o final deste século [HUI97] [IBM95] [GRA96].

O crescimento exponencial da *Internet* de forma alguma foi planejado. O seu projeto inicial não contava ainda com a hierarquia de endereços que conhecemos hoje. No início, o endereçamento era composto por 32 bits, onde 8 bits indicavam a rede e os demais 24 bits, indicavam o *host*. A separação em classes A, B, C, para grandes, médias e pequenas redes respectivamente, foi adicionada mais tarde, quando foi percebido que poderiam haver muito mais que 256 redes na *Internet*.

Três grandes problemas foram gerados devido a esse crescimento da *Internet*:

- exaustão dos endereços de classe B;
- explosão das tabelas de roteamento;
- escassez significativa de endereços IP.

Para a solução imediata dos problemas citados anteriormente, foram desenvolvidos o CIDR (*Classless Inter-Domain Routing*) (ver Seção 3.6.3) e o NAT (*Network Address Translator*) (ver Seção 3.6.4), permitindo, dessa forma, que a expansão da *Internet* fosse mantida até o desenvolvimento de um novo protocolo IP (*Internet Protocol*).

4.1 Histórico

No final da década de 80 já se evidenciava a preocupação com o crescimento da *Internet*. Em 1988, Robert Ullmann lançou, informalmente, uma proposta para o aperfeiçoamento da camada IP. Este protocolo foi formalmente conhecido em 1992 com o nome de TP/IX (IPv7), e constituíu-se posteriormente, em uma das três propostas avaliadas pela IETF (*Internet Engineering Task Force*) para se tornar a nova versão do IP (foi submetido com o nome de CATNIP). Paralelamente a isto, em dezembro de 1986, a ISO (*International Standard Organization*) apresentou o texto final do seu protocolo para o fornecimento de um Serviço de Rede sem Conexão conhecido como CLNP (*Connection-Less Network Protocol*). O CLNP faz uso de endereços NSAP (*Network Service Acess Point*) com tamanho variável de até 20 octetos, permitindo uma gama enorme de endereços possíveis.

O ano de 1992 pode ser considerado, efetivamente, como o ano embrionário da Nova Geração do Protocolo *Internet*. Em fevereiro, foram apresentadas quatro propostas à IETF: Nimrod , IP Encaps , CNAT , Simple CLNP.

Apenas dois meses após o lançamento destas propostas, ocorreram algumas mudanças significativas. O Simple CLNP evolui para o TUBA (*TCP/UDP over Bigger Address*) e o IP Encaps evolui para o IPAE (*IP Address Encapsulation*) .

Por volta de dezembro de 1992, mais três propostas foram apresentadas à IETF: PIP (*Paul's Internet Protocol),* SIP (*Simple Internet Protocol*) e TP/IX .

Alguns projetos se agruparam a outros projetos, fortalecendo estes protocolos. Em dezembro de 1993, o IPAE se uniu ao SIP, havendo a manutenção do nome SIP. Posteriormente o PIP também se agregou ao SIP, formando o SIPP (*Simple Internet Protocol Plus).* Neste mesmo período o TP/IX teve seu nome mudado para CATNIP (*Common Architecture for the Internet*).

Em dezembro de 1993, a IETF, lançou o RFC 1550 [BRA93], oficializando o envio de propostas para a efetivação de um novo protocolo IP.

Os aspectos relevantes (requisitos) avaliados pela IETF no processo de seleção das propostas foram os seguintes:

- expansibilidade: qual seria a estimativa para a expansão das redes no futuro;
- tempo para implantação: definição de estimativas para a seleção, desenvolvimento e efetiva utilização do IPng;
- transição: transição do IPv4 para o IPv6. O que deveria ser considerado para que o processo de transição seja o mais simplificado e eficiente possível, possibilitando a migração para o novo protocolo, sem "traumas" para atual estrutura da *Internet*;
- segurança: que tipo e nível de segurança serão desejáveis para o ambientes de redes futuros;
- configuração, administração e operação: o que a nova geração do IP deve possuir para facilitar a administração das redes por seus operadores.
- mobilidade: quais as características devem ser empregadas para que a nova geração do IP possibilite *hosts* móveis, ou seja, que um *host* possa ser conectado em qualquer *link*, sem que seja necessário modificar a sua configuração;
- reserva de recursos e fluxo: devido ao crescimento no número de processos em tempo-real, devem ser definidos os métodos a serem empregados para que os recursos possam ser reservados e liberados em momentos críticos;
- política baseada em roteamento: que tipo de política de roteamento será empregada, na nova geração do protocolo *Internet*;
- flexibilidade na topologia: o modelo geral de topologia empregado deve ser mantido ou é necessário estabelecer o emprego de restrições topológicas;
- aplicabilidade: a que tipo de ambiente a nova geração de IP se aplica;
- serviço de datagrama: se os datagramas deverão ou não serem roteados *hop-a-hop*;
- contabilização: deve-se definir se o IPng terá contabilização e sua amplitude de atuação;

- suporte aos meios de comunicação: deve- se definir se o IPng manter ou não a mesma flexibilidade que o IPv4 diante das tecnologias de comunicação;
- robustez e tolerância a falhas;
- tecnologias que impulsionam a *Internet*: deve-se definir de que modo as tecnologias emergentes devem influenciar a nova geração do IP.

Das propostas enviadas, 3 delas foram consideradas para serem avaliadas: TUBA, CATNIP e SIPP.

4.1.1 As Propostas Consideradas pela IETF

4.1.1.1 TUBA (*TCP/UDP over Bigger Address*)

A proposta do TUBA está totalmente centrada na substituição do protocolo IP pelo CLNP da ISO. Os protocolos da camada de transporte (TCP, UDP), bem como as tradicionais aplicações devem atuar no topo do CLNP.

O esforço do TUBA concentra-se em permitir a expansão da capacidade de roteamento de pacotes através da utilização de endereços que suportem mais níveis de hierarquia do que o sistema de endereçamento do IPv4.

4.1.1.2 CATNIP (*Common Architecture for the Internet*)

O CATNIP foi projetado para ser um protocolo de convergência, integrando CNLP, IP e IPX. Através de suas definições, todos os protocolos da camada de transporte existentes, usados em serviços sem conexão operariam sobre qualquer protocolo da camada de rede existente.

4.1.1.3 SIPP (*Simple Internet Protocol Plus*)

De certo modo, as duas propostas apresentadas anteriormente se caracterizam por exigirem mudanças significativas na camada de rede. Por este motivo, dificilmente podería-se considerá-las como uma evolução natural do protocolo IP.

O protocolo SIPP surgiu exatamente como uma evolução natural do IP. Os métodos que eram funcionais na versão 4 se mantiveram e aqueles não funcionais foram excluídos. Outras características foram introduzidas para garantir a expansão e a evolução da *Internet*, diante do surgimento de novas tecnologias.

As principais características do protocolo SIPP são:

- o tamanho do endereço IP é expandido de 32 bits para 64 bits, permitindo mais níveis de hierarquia de endereçamento, bem como aumentando significativamente o número de nós endereçáveis;
- simplificação do formato do cabeçalho;
- mudanças nas opções do cabeçalho;
- capacidade de rotular os pacotes, permitindo assim que a qualidade do serviço seja definida. Dessa forma, serviços não padrões ou serviços em tempo real podem ser requisitados;
- autenticação, integridade dos dados, e confidencialidade são fornecidos através de extensões do cabeçalho padrão.

4.1.1.4 Avaliação das Propostas do IPng

Os protocolos TUBA, CATNIP e SIPP, foram rigorosamente avaliados pela comissão IPng da IETF, através de teleconferências e de listas de correio eletrônico.

O CATNIP foi considerado inadequado por não apresentar uma especificação completa. Muitos dos quesitos desejáveis apresentados anteriormente não

foram sequer referenciados. Apesar disto, foi considerado o protocolo que apresentou as idéias mais inovadoras.

Os revisores do TUBA consideraram este protocolo importante por permitir uma futura convergência entre os padrões de rede ISO e IETF. Como ele se fundamenta no CLNP, poderia ser aproveitado o fato de que muitos roteadores na *Internet* trabalham com este protocolo, não havendo necessidade que esta base de roteadores tivesse que definir nova implementação. Algumas características requisitadas não foram definidas ou foram fracamente definidas, o que sugere algumas modificações no CLNP. Entretanto, muitas dúvidas surgiram a respeito de qual instituição teria os direitos sobre o novo protocolo, uma vez que o protocolo originário pertence a ISO e a ela pertence o direito de alteração. Este motivo foi o suficiente para o abandono desta proposta, uma vez que a IETF tem a intenção de ser totalmente soberana na padronização dos protocolos da *Internet.*

De todos os grupos de trabalho envolvidos com a nova geração do IP, o SIPP foi o que apresentou a maior dinâmica e coerência com as expectativas da IETF. Uma série de documentos foram confeccionados abordando quase todos os aspectos importantes considerados na definição do novo protocolo. Entretanto, algumas questões como a transição do protocolo, os aspectos de segurança, a autoconfiguração, foram consideradas insatisfatórias. Por este motivo, houve a necessidade de uma completa revisão. Outro ponto considerado negativo, foi o tamanho de 64 bits do endereço IP, considerado insuficiente. Após a primeira revisão feita pela IETF, o grupo do SIPP entregou uma proposta revisada que corrige as principais falhas da especificação original, tendo sido então escolhido para ser o substituto do IPv4. O novo protocolo teria 128 bits de endereçamento e se chamaria IPv6 .

4.2 Formato do Cabeçalho IPv6

O cabeçalho IP da versão 6 tornou-se mais simples em comparação ao IPv4. Todos os campos subutilizados foram removidos e aqueles que permaneceram foram revisados.

O novo cabeçalho conta com 8 campos enquanto que a versão anterior continha 12 campos fixos e algumas opções. O único campo que se manteve intacto foi o campo da versão. Os campos removidos foram os seguintes: tamanho do cabeçalho, tipo de serviço, identificação, *flags*, *off set*, e checagem de erro do cabeçalho. Três campos foram renomeados e redefinidos: tipo de protocolo, tamanho do pacote e tempo de vida. Dois campos foram introduzidos: classe e rótulo de fluxo. O mecanismo de opções foi revisado, no seu lugar foram definidas extensões do cabeçalho (ver Seção 3.2).

O tamanho total do cabeçalho IPv6 será sempre de 40 *bytes*, o dobro do tamanho do cabeçalho básico do IPv4.

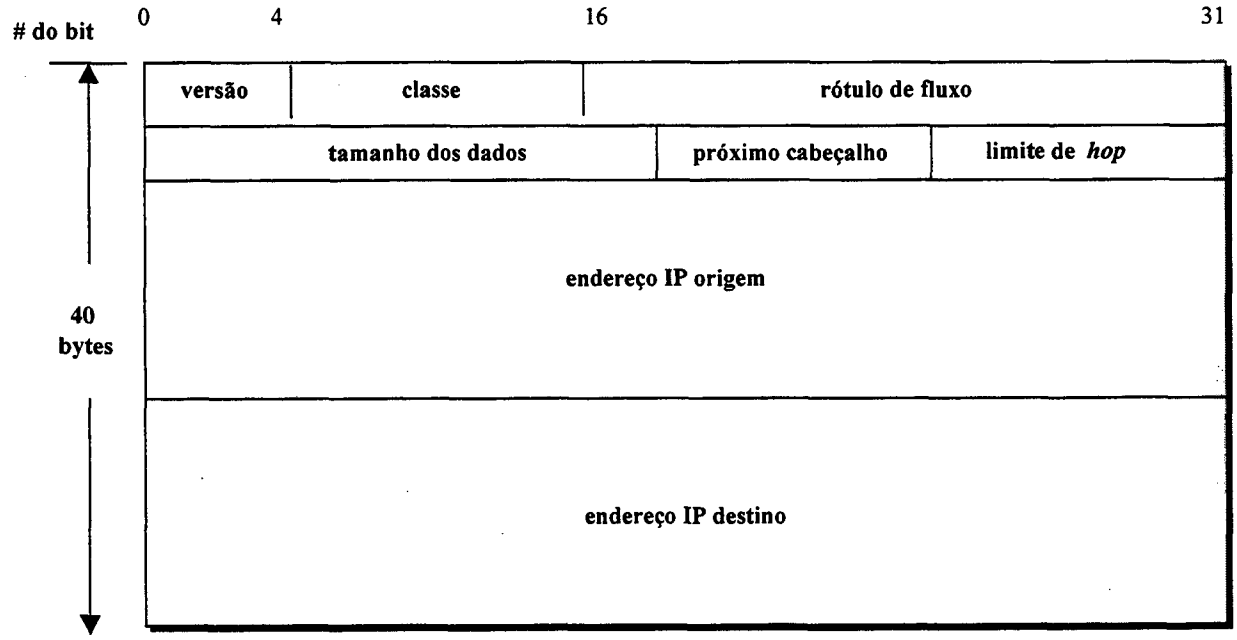


Figura 4. 1 Cabeçalho IP

A Figura 4.1 apresenta o formato do datagrama IPv6 onde:

- versão: Define o versão do protocolo IP (4 bits);
- classe: bits de controle de tráfego em tempo-real (8 bits). Campo introduzido nesta versão. Opera juntamente com o rótulo de fluxo no gerenciamento de tráfego (ver Seção 2.5.6);
- rótulo de fluxo: opera no gerenciamento de tráfego (20 bits);

- tamanho dos dados: define o tamanho do pacote a partir do cabeçalho IPv6, em octetos (16 bits). Substitui o campo total do IPv4;
- próximo cabeçalho: identifica o tipo de cabeçalho imediatamente após o cabeçalho IPv6 (8 bits). Nesse caso pode ser o cabeçalho de extensão ou o cabeçalho de um protocolo da camada de transporte;
- limite de *hop*: A cada nó intermediário que o pacote passar, deve ser decrementado em 1. O pacote será descartado se o valor atingir zero. Substitui o campo tempo de vida (TTL) do IPv4. Baseia-se na contagem do número de *hops* e não no número em segundos (8 bits);
- endereço IP origem: endereço da máquina que origina o pacote (128 bits);
- endereço IP destino: Endereço da máquina destino (128 bits).

4.3 Tamanho do Datagrama IPv6

O IPv6 requer que todo meio físico na *Internet* suporte um MTU (*Maximum Transfer Unit*) igual ou superior a 576 bytes . Nos *links* onde esta característica não se apresente, a fragmentação e a recomposição deverão ser fornecidas pela camada imediatamente inferior ao IPv6.

Atualmente, os *links* que compõem a *Internet* possuem uma velocidade e confiabilidade considerável. Existe uma tendência que estas características venham aumentar cada vez mais. Normalmente, o tamanho máximo de um pacote é determinado pela taxa de transmissão e de erro. Nas redes mais velozes e confiáveis, é extremamente desejável que o tamanho do pacote apresentado pelo meio físico seja maior, pois diminui a possibilidade de *overhead*, uma vez que uma quantidade menor de datagramas será enviada, diminuindo assim as informações adicionais de cabeçalhos que são agregadas a cada um dos fragmentos que compõe o pacote. O aumento do tamanho do pacote também reduz a requisição da CPU (*Central Processing Unit*) nos *hosts* . Dessa

forma para que o IPv6 caminhe junto com estes avanços tecnológicos, foi definido o tamanho do MTU em 576 octetos.

A especificação do protocolo recomenda que os nós implementem *Path MTU Discovery*, que permite que o MTU suportado por todos os links seja dinamicamente determinado, e dessa forma, o nó origem enviará apenas datagramas que não excedam o *Path MTU*, ou seja um tamanho de pacote aceitável por todos os meios (Ethernet, FDDI, etc.) naquele caminho. A importância da implementação deste procedimento está no fato de que os datagramas alcançarão o destino sem a necessidade de serem fragmentados.

4.4 Extensões do Cabeçalho IPv6

O campo opções do IPv4, surgiu para permitir que alguns datagramas fossem transportados recebendo uma avaliação e um processamento distinto dos datagramas comuns. Aspectos de segurança, roteamento, etc. são abordadas neste campo. Um ponto importante a ser considerado é que o campo opções do cabeçalho IPv4 deve ser avaliado por todos os nós envolvidos no caminho de entrega do pacote, determinando que o tempo de entrega de um pacote não padrão seja mais elevado que o normal. Por esta razão o campo ficou em desuso.

Questões como segurança, roteamento, fragmentação, foram destacadas no IPv6. São pontos não funcionais no IPv4, mas importantes para o próprio futuro da *Internet*.

A forma encontrada no IPv6, para o transporte de informações opcionais, foi a codificação desta informação, em cabeçalhos separados. Foi definido um número pequeno deste tipo de cabeçalho, entretanto, existe a flexibilidade no aumento deste número.

No transporte de um pacote IPv6 pode haver zero ou mais cabeçalhos de extensão, que serão examinados ou processados apenas pelos nós origem e destino. O nós intermediários não avaliam estes cabeçalhos, garantindo que o transporte

seja o mais leve possível. Existe apenas uma exceção, o cabeçalho *hop-a-hop* que deve ser examinado e processado por todos os nós ao longo do caminho de entrega do pacote.

Existem seis cabeçalhos de extensão definidos:

- cabeçalho *hop-a-hop*: opções especiais que requerem processamento *hop-a-hop*. Verificada em todos os roteadores ao longo do caminho de entrega do pacote;
- cabeçalho de roteamento: fornece extensões ao roteamento;
- cabeçalho de fragmentação: fragmentação e montagem dos datagramas;
- cabeçalho de autenticação: integridade e autenticação;
- cabeçalho de encapsulamento: criptografia, confidencialidade;
- cabeçalho fim-a-fim: informação opcional examinada pelo nó destino.

4.5 Arquitetura de Endereçamento

Exatamente como na versão IPv4, existe uma relação direta entre interface e endereço IP, ou seja, cada endereço IP identifica uma interface. Assim um *host multi-homed*[1], possui vários endereços IP. Neste novo padrão são definidos 3 tipos de endereçamento: *unicast, anycast, multicast*. O modo de endereçamento *broadcast* foi eliminado, sendo suas funções atribuídas ao endereçamento *multicast*.

[1] Pertence a mais de uma subrede

4.5.1 Tipos de Endereçamento

Os tipos de endereço definidos pelo protocolo IPv6 são:

- *Unicast*: endereço que identifica uma única interface. Um pacote enviado para um endereço *unicast* é entregue para a interface identificada por aquele endereço;
- *multicast*: identifica um conjunto de interfaces, normalmente pertencentes a diferentes nós. Um pacote enviado a um endereço multicast será entregue a todos os membros do grupo;
- *anycast*: similar ao endereçamento *multicast*, o *anycast* identifica também um grupo de interfaces. Um pacote enviado a um endereço *anycast* será entregue a uma das interfaces do grupo, aquela que se apresente mais próxima, seguindo os critérios métricos do protocolo de roteamento.

4.5.2 Representação dos Endereços

Um endereço IPv6 é composto por 128 bits, representados por 8 campos de 16 bits separados por dois pontos (:). Cada um dos campos é representado por 4 dígitos hexadecimais.

Existem três formas padrões de representação dos endereços IPv6:

- Formato x:x:x:x:x:x:x:x, sendo x um valor hexadecimal com tamanho de 16 bits

Exemplo: FEDC:BA98:7654:3210:FEDC:BA98:7654:3210

1080:0000:0000:0000:0008:0800:200C:41B5

É permitido reduzir a forma da representação quando o campo possuir zeros que não alterem o valor do hexadecimal. A forma reduzida para o segundo exemplo visto acima será: 1080:0:0:0:8:800:200c:41B5

- Alguns métodos de alocação de endereço IPv6 utilizam múltiplos grupos com valores zero. Para facilitar a escrita a seguinte sintaxe foi adotada: **::**
  Exemplo: 1080:0:0:0:0:8:800:200C:41B5
  Utilizando a sintaxe :: - 1080::8:800:200C:41B5
  É importante observar que será permitido que este tipo de notação (**::**) seja utilizado uma única vez na representação do endereço.
  Exemplo de valor não válido: 0:0:0:0:FFCC:0:0:3210 - ::FFCC::3210

- Formato x:x:x:x:x:x.d.d.d.d. Forma alternativa de representação utilizada na transição de IPv4 para IPv6. O caracteres x representam os valores hexadecimais de 16 bits e os caracteres d representam os valore decimais de 8 bits, mantendo a notação original do IPv4.
  Exemplo: 0:0:0:0:0:0:0:150.162.1.3
  Forma reduzida ::150.162.1.3

4.5.3 Prefixo dos Endereços

Uma das grande preocupações do projeto IPv6 está centrada no tamanho do endereço IPv6. Este tamanho deve ser suficiente para que se possa definir uma gama variada de tipos de endereço. Endereços estes, que possam representar diferentes situações, diferentes protocolos, diferentes objetivos.

Até o momento uma alocação inicial de endereçamento foi definida (ver Figura 4.2). Através do campo prefixo do formato, os diversos tipos de endereço são classificados.

Alocação	Prefixo do formato (binário)
reservado	0000 0000
não atribuído	0000 0001
reservado para alocação NSAP	0000 001
reservado para alocação IPX	0000 010
não atribuído	0000 011
não atribuído	0000 1
não atribuído	0001
não atribuído	001
endereços *unicast* de agregação global	010
não atribuído	011
reservado para endereços *unicast* baseados na geografia	100
não atribuído	101
não atribuído	110
não atribuído	1110
não atribuído	1111 0
não atribuído	1111 10
não atribuído	1111 110
não atribuído	1111 1110 0
endereços *link-local*	1111 1110 10
endereços *site-local*	1111 1110 11
endereços *multicast*	1111 1111

Figura 4. 2 Alocação Inicial de Endereços para IPv6

A representação textual do prefixo dos endereços é da seguinte forma: **endereço-IPv6/tamanho-do-prefixo,** onde:

- endereço-IPv6: é um endereço IPv6;
- tamanho-do-prefixo: valor decimal que representa quantos dos bits de mais alta ordem (posicionados mais à esquerda) definem o prefixo do endereço.

Exemplo: 12AB:0:0:CD30:0:0:0:0/60

12AB:0:0:CD3: define o tipo de endereço

Os endereços *unicast* são distinguidos dos endereços *multicast* pelo valor do octeto de mais alta ordem do endereço. Um valor FF (11111111) identifica um endereço *multicast*, qualquer outro valor identifica um endereço *unicast*. Endereços *anycast* estão no espaço de endereçamento *unicast*, e, sintaticamente não são distingüíveis de um endereço *unicast*.

4.5.4 Endereços *Unicast* de Agregação Global

Este formato de endereço está definido para suportar agregação baseada nos provedores de acesso à *Internet* e um tipo de agregação chamada de troca. Este tipo de combinação permitirá eficiente agregação de roteamento para os *hosts* que estão conectados diretamente a provedores de acesso ou quem estiver conectados a trocas.

3	13	32	16	64 bits
010	TLA	NLA	SLA	Interface ID

Figura 4. 3 Formato dos Endereços *Unicast* de Agregação Global

O formato do endereço é apresentado na Figura 4.3 onde:

- 010: Prefixo para endereços de agregação global;
- TLA ID: identificador de agregação *top-level*;
- NLA ID: Identificador de agregação *next-level*;
- SLA ID: Identificador de agregação *site-level*;
- INTERFACE ID: identificador da interface .

4.5.5 Formatos de Endereços *Unicast* Especiais

Existem 4 tipos de endereços *unicast* especiais:

- endereço sem especificação: composto por 16 bits nulos. Seu valor é ::, sendo resultado da simplificação de 0:0:0:0:0:0:0:0. Pode ser utilizado apenas como endereço origem em uma estação que ainda não foi configurada com um endereço regular. Nunca será utilizado como endereço destino e não poderá estar associado a uma interface;

- endereço loopback: o seu valor é ::1, forma reduzida de 0:0:0:0:0:0:0:1. Pode ser utilizado quando um nó enviar um pacote IPv6 para si mesmo. Não poderá ser alocado a uma interface;

- endereço baseado no IPv4: 96 bits zero e 32 bits de endereço IPv4. A notação utilizada é ::d.d.d.d . Como exemplo, uma máquina com endereço IPv4 igual a 150.162.1.4, com a notação de transição seu novo endereço será ::150.162.1.4. Este tipo de endereço será utilizado nos períodos de transição IPv4/IPv6;

- endereços *site-local*: endereços definidos para utilização interna. Ou seja para as redes que não intencionam o acesso à *Internet* Global. Os roteadores não devem repassar qualquer pacote cujo endereço origem seja do tipo *site-local*, ou cujos endereços destinos não pertençam a rede local. Um endereço *site-local*, constará de de um prefixo (FEC0), um grupo de zeros, a identificação da subrede, e o identificador da interface;

- endereços *link-local*: estes endereços são definidos dentro de um *link* e podem ser usados por nós conectados ao mesmo meio ou mesma rede local. Datagramas enviados a estes endereços nunca serão repassados pelos

roteadores. Um endereço *link-local* é composto de um prefixo (FE80), um grupo de zeros, e um identificador de interface. Maiores informações são encontradas na Seção 7.2.2.

4.6 Qualidade de Serviço

Sempre que houver a necessidade de comunicações multimídia em tempo real, é requisitado um controle de fluxo adequado. Estes tipos de comunicações são emergentes, e possivelmente serão bastante utilizadas. O IPv6 tenta garantir a qualidade de serviço através dos campos classe e rótulo de fluxo, presentes no cabeçalho padrão. Estes campos foram projetados para que um *host* possa identificar aqueles datagramas que necessitam ter um tratamento especial.

O campo que rotula o fluxo pode ser usado pelo nó origem para identificar aqueles datagramas ao quais reserva-se um tratamento diferenciado. O campo classe permite que datagramas sejam identificados e diferenciados, possibilitando que sejam priorizados segundo sua classe de tráfego de datagramas IPv6. Experimentos ainda estão sendo feitos para a definição destas classes de tráfego.

4.7 Transição

Para que a implantação e o desenvolvimento do IPv6 ocorra de uma forma gradual, foram definidos mecanismos que mantivessem a compatibilidade com o IPv4. Mecanismos foram formulados para que *hosts* e roteadores IPv6 sejam compatíveis com *hosts* e roteadores IPv4.

4.7.1 Mecanismos de Transição

Os mecanismos de transição adotados pelo IPv6 são os seguintes:

- Camada IP Dupla: suporte completo às duas versões, IPv4 e IPv6;
- tunelamento IPv6 sobre IPv4: os datagramas IPv6 devem ser encapsulados dentro dos cabeçalhos IPv4, sendo dessa forma transportados pela estrutura de roteamento IPv4.

4.7.2 Tipos de Nós

Os tipos de nós definidos pelo IPv6 são os seguintes:

- nó IPv4: *host* ou roteador que implementa apenas IPv4. Não entende IPv6;
- nó IPv6: *host* ou roteador que implementa apenas IPv6. Não entende IPv4;
- nó IPv6/IPv4: *host* ou roteador que implementa IPv4 e IPv6. Entende IPv4 e IPv6.

4.7.3 Camada IP Dupla

Para que nós IPv6 se mantenham totalmente compatíveis com nós IPv4, foi definido que possuíssem as duas implementações, IPv4 e IPv6.

4.7.4 Configuração do Endereço

Os nós IPv4/IPv6 devem ser configurados com endereços IPv6 e IPv4. Nos nós IPv6, que utilizam o mecanismo de tunelamento, deve ser utilizado o endereço baseado no IPv4 (endereço *unicast* especial).

4.8 Suporte à Mobilidade

O cenário que se apresenta atualmente, diante da necessidade de transferir um nó para um outro *link* é o seguinte: o nó que é conectado a um novo meio físico deve ser necessariamente reconfigurado. Receber um novo endereço IP e possivelmente um novo nome. Com o advento do DHCP (*Dynamic Host Configuration Protocol*), o trabalho de configuração ficou facilitado. Entretanto, existem algumas situações que devem ser consideradas:

- computador não estar configurado para receber configuração automática via DHCP, necessitando dessa forma intervenção manual do administrador da rede;
- normalmente acesso a determinadas máquinas são permitidas para aquele nó, somente quando se apresentar com sua configuração original. Por exemplo, um nó móvel, era configurado com IP 150.162.9.200 e nome jussara.apl.npd.ufsc.br. Este computador não tem servidor de *e-mail*. Portanto, tem que acessar este serviço através de um servidor. Por questões de confiabilidade, este servidor de *e-mail*, contém uma tabela de *hosts*, cujo acesso ao serviço de *e-mail* é permitido. A tabela é constituída por endereços ou subredes IP. Uma vez que uma máquina cliente mude seu endereço IP, terá seu acesso automaticamente negado.

Para evitar situações como as descritas acima, é necessário dotar o protocolo com mecanismos que permitem mobilidade. O mecanismo que permite mobilidade ao *host* IPv6 é o seguinte: primeiramente é importante salientar que sem suporte específico para mobilidade em IPv6, datagramas destinados a um nó móvel não serão capazes de serem alcançados enquanto ele não estiver conectado ao seu *home link* (meio físico onde está normalmente conectado).

Todos os nós móveis, sempre serão identificados e endereçáveis por seu *home address*, não importando a que *link* na *Internet* estará conectado. O *home address*, nada mais é, que o endereço atribuído ao nó móvel, com o prefixo da sua subrede original (*home subnet*) na *Internet*. Quando o nó móvel estiver conectado ao seu *home link*, não será necessária a intervenção de qualquer outro nó para que ele estabeleça a comunicação com os demais nós na *Internet*, com exceção das atividades de roteamento. Um nó móvel, ao se conectar em um *link* qualquer, que não seja o seu *home link*, será unicamente endereçável através de um endereço chamado *care-of-address*. Este novo endereço será conhecido apenas pelo seu *home agent* (roteador de seu *home link)*. Os demais nós na *Internet* manterão comunicação através do endereço original. Um *care-of-address* é um endereço IP associado ao nó móvel enquanto estiver conectado a um *foreign link*. Um nó móvel normalmente adquire seu endereço IP através de autoconfiguração, de acordo com os métodos IPv6 de descoberta de vizinho. O nó móvel ao receber seu endereço IP (*foreign ip*) avisa ao seu roteador (do *home link*). Automaticamente, o roteador redirecionará os datagramas com endereço destino apontados ao nó móvel para seu novo endereço.

4.9 Segurança

A *Internet* tem sofrido problemas de segurança, devido a uma série de fragilidades inerentes ao próprio protocolo TCP/IP. No IPv4 verifica-se a ausência de mecanismos efetivos que garantam a privacidade e a autenticação a nível de camada de rede.

Atualmente existem meios de reforçar a segurança na camada IP, adicionando *software* e utilizando técnicas para reduzir os ataques. Porém a segurança não é nativa do protocolo IP, trata-se de um sistema deficiente, com muitas adaptações, motivo pelo qual torna-se mais caro e complexo (a cada novo problema de segurança encontrado surge um novo software). Além disso, não basta que uma das partes comunicantes tenha implementações adicionais de segurança, se o meio e a outra parte envolvida não são seguros.

A maioria das técnicas desenvolvidas não garantem total segurança, e sim reduzem os máximo as chances de ataque.

A nova versão do protocolo IP implementa segurança corrigindo as deficiências da versão anterior e reduzindo a necessidade de adaptação de técnicas para atenuar as falhas.

O IPv6 implementa dois cabeçalhos de extensões que lidam exatamente com questões referentes a segurança.

O cabeçalho de autenticação provê autenticidade e integridade, porém não garante confidencialidade. A confidencialidade é fornecida através do cabeçalho de encapsulamento, através de métodos criptográficos.

4.10 Relação com as Camadas Superiores

As alterações que devem ser feitas a nível de camadas de transporte e aplicação são mínimas. Basicamente, estão centradas na alteração do formato do novo

endereço IP. Com relação a camada de transporte, outras alterações devem ser feitas para trabalharem com a nova versão do protocolo ICMP que estabelece novos tipos de mensagens.

Um ponto interessante a ser abordado, é que a camada imediatamente superior ao protocolo IPv6 deve garantir a checagem de erros (*checksum error*), pois o IPv6 aboliu este tipo de verificação, que inclusive na versão anterior é bastante limitada (checa apenas o cabeçalho IP).

4.11 Serviço de Domínio de Nomes (*Domain Name Service-DNS*)

O serviço de nomes, mantém a mesma características encontradas no IPv4 (ver Seção 3.8). Com relação ao mapeamento direto, foi introduzido o *resource record* "AAAA". Para o endereçamento reverso, o sufixo .in-addr.arpa. foi substituído pelo sufixo .IP6.INT. para refletir melhor a natureza internacional da *Internet*.

4.12 Autoconfiguração

Uma máquina se autoconfigurará quando descobrir e registrar automaticamente os parâmetros que ela necessitar para se conectar à *Internet* [HUI97].

O processo de autoconfiguração deve constar dos seguintes ítens:

- criar um endereço *link-local* e certificar-se que é único no *link*;
- determinar qual deve ser o método de autoconfiguração e quais parâmetros devem ser autoconfiguráveis.

Os métodos de autoconfiguração são os seguintes:

Método *Stateless* (*Stateless Mode*) [THO96]: Uma máquina ao se conectar à rede, iniciará o processo de autoconfiguração através da atribuição automática de um endereço *link-local* à sua interface. O endereço *link-local*, como visto anteriormente, é composto pelo prefixo já definido FE80:0:0:0 mais o identificador de sua interface. O endereço só será atribuído a interface se não tiver réplica na rede. Depois que um endereço *link-local* foi atribuído, a próxima fase determinará se haverá necessidade de uma autoconfiguração *stateful* ou não. Uma vez que o método *stateless* apenas atribui endereço *link-local*, ele permite apenas uma comunicação restrita às máquinas conectadas ao mesmo *link*. A determinação será feita por um roteador. De tempos em tempos, o roteador envia ao *link* avisos que especificam que tipo de autoconfiguração um *host* deve fazer. O aviso do roteador contém 2 campos. O primeiro campo indica que tipo de autoconfiguração deve ser adotado, e o segundo, determina quais outros tipos de parâmetros devem ser obtidos, caso a configuração optada seja *stateless*. Se não existirem roteadores, o método de autoconfiguração *stateful* deve ser invocado.

Método *Stateful* (***Stateful Mode***): O único método *stateful*, de autoconfiguração do IPv6, até então definido, é o DHCPv6 (*Dynamic Host Configuration Protocol for IPv6*). Neste método, o *host* autoconfigurável, receberá um endereço IP completo, passível de acesso através da *Internet*. Outros parâmetros, como por exemplo, endereço da máquina servidora de DNS, poderão também ser configurados.

Uma descrição completa e detalhada dos protocolos de autoconfiguração para o IPv6 será apresentada no capítulo 8.

4.13 Roteamento

O roteamento do protocolo IPv6 é semelhante ao modelo CIDR, do protocolo IPv4. A diferença está no tamanho do endereço IP, que aumentou de 32 bits para 128 bits. Segundo projetistas do IPv6 [HUI97][HID96], com algumas alterações, todos os algoritmos de roteamento (OSPF, RIP, IDRP, IS-IS, BGP, IGRP, etc.) utilizados na versão 4 do IP poderão ser utilizados na versão 6.

Atualmente, os protocolos RIP e BGP-4, já tem seus padrões propostos em RFCs, para a IPv6. Os protocolos OSPF e IGRP, estão em fase de desenvolvimento.

Como vimos na Seção 3.6.2, existem dois tipos de protocolo de roteamento. Os protocolos de roteamento interiores e os protocolos de roteamento exteriores:

- protocolos de roteamento interiores: "os protocolos de roteamento interiores, são utilizados para computar e manter conectividade de um domínio de roteamento ou um sistema autônomo" [HUI97]. A versões deste protocolos (OSPF, RIP, IS-IS, IGRP) deverão ser apenas adaptadas ao novo protocolo;

- protocolos de roteamento exteriores: os protocolos de roteamento exteriores (EGP, BGP, BGP-4), têm como função a troca de informações de roteamento entre Sistemas Autônomos. Como visto anteriormente, o roteamento IPv6 é baseado no CIDR. É interessante notar que, se a estrutura de roteamento utilizada no IPv6 não utilizasse o modelo CIDR (utilizada pelo BGP-4), provavelmente, o aumento do tamanho do endereço de 32 bits para 128 bits, resolveria apenas o problema do número reduzido de redes e endereços IP. Não resolvendo o problema da explosão nas tabelas de roteamento. O modelo CIDR é baseado na agregação de várias entradas da tabela de roteamento. A agregação, no roteamento IPv6, acontece através de sua hierarquia. As rotas são agregadas de acordo com a estrutura de provedores da *Internet*. A idéia original era agregar por continente, país,

região, cidade, etc. Porém, a estrutura atual da *Internet*, não reflete este tipo de hierarquia. Sabe-se que em um mesmo país existem vários fornecedores de acesso à *Internet*. E seguindo esta estrutura é que se fará a agregação das rotas na *Internet*.

4.14 ICMPv6

Por ser projetado para atender as necessidades do IPv6 o ICMPv6 não é compatível com sua versão anterior. O que é subutilizado na versão 4 foi removido tornando-se mais completo por incorporar funções de *multicast* do protocolo IGMP (*Internet Group Membership Protocol*) do IPv4.

4.14.1 Formato da Mensagem ICMPv6

Apesar do ICMP ser um protocolo que pertence a camada de rede, ele envia sua mensagens dentro do datagrama IP. O valor 58 do campo próximo cabeçalho no cabeçalho IPv6 indica uma mensagem ICMPv6.

Tipo	Código	controle de erros
corpo da mensagem		

Figura 4. 4 Formato Geral das Mensagens ICMPv6

A Figura 4.4 apresenta o formato geral das mensagens ICMPv6 onde:

- tipo: valor inteiro que indica o tipo da mensagem. Valores de 1 até 127 indicam mensagens de erro, valores de 128 à 255 indicam mensagens informacionais (8 bits);
- código: está vinculado ao tipo de mensagem. Atua determinando o grau de especificação (nível de granularidade) da mensagem (8 bits);
- controle de erro: é utilizado para detectar a corrupção nos dados das mensagens ICMPv6 (16 bits);
- corpo da mensagem: contém a mensagem propriamente dita.

As mensagens que serão enviadas pelo ICMPv6 são agrupadas em 2 classes: mensagens de erro e mensagens informacionais.

4.14.2 Mensagens de Erro do ICMPv6

1. Destino inalcançável: a mensagem de destino inalcançável deverá ser gerada por um roteador, ou pela camada IPv6 do *host* origem, em resposta a um datagrama que não pode ser entregue ao *host* destino. As possíveis razões para que o datagrama não alcançe seu destino são:

   - impossibilidade de repasse do datagrama, por não haver referência do IP na tabela de roteamento (ausência de rota específica e ausência de rota *default*);
   - proibição administrativa;
   - qualquer outra razão que não seja congestionamento da rede. Uma mensagem ICMPv6 não deverá ser gerada se um pacote não puder ser entregue devido a problemas de congestionamento da rede [CON97].

2. Pacote muito grande: uma mensagem de pacote muito grande deverá ser enviada por um roteador em reposta a um pacote que não pode ser repassado por seu tamanho ser maior que a MTU (*Maximum Transmission Unit*) definido pela tecnologia de rede

do *link* para onde o ele seria enviado. Este tipo de mensagem será utilizada no processo de descoberta da MTU (*Path MTU Discovery*)

3. Tempo excedido: Este tipo de mensagem é enviada quando o roteador que está processando o datagrama verifica que o campo limite de *hop* possui um valor igual a zero, sendo então, necessário destacar este datagrama;

4. Problema com parâmetros: durante o processamento de um datagrama IPv6, poderão ser encontrados problemas em alguns campos dos cabeçalhos IPv6 ou do cabeçalhos de extensão, impedindo o processamento deste datagrama. Este tipo de mensagem deverá ser enviada ao *host* que originou o datagrama, indicando o tipo e localização do problema. Por exemplo: valor incorreto do campo próximo cabeçalho.

4.14.3 Mensagens Informacionais do ICMPv6

Echo requeste echo reply : estes tipos de mensagens são utilizadas para o diagnóstico de redes, o ICMP determina se existe a comunicação entre dois *hosts*. O *host* origem utiliza o protocolo ICMP para enviar uma mensagem utilizando *echo*, e esta mensagem deve ser devolvida através de um *echo reply* pelo *host* destino. Um exemplo de aplicação que utiliza este recurso é o ping.

Protocolo *Neighbor Discovery*: este protocolo implementa 5 mensagens ICMPv6 utilizadas para troca de informações entre nós vizinhos que são fundamentais no processo de autoconfiguração *stateless* do IPv6 (ver Seção 7.1).

4.15 *Backbone 6Bone*

O *Backbone 6bone* faz parte do projeto *IP Next Generation* conduzido pela IETF. Ele é uma parte essencial no projeto de transição IPv4/IPv6. Tem como objetivo a formação de uma rede de amplitude mundial que possa transportar datagramas IPv6.

O *6bone* é uma rede virtual, formada por ilhas de redes que suportam IPv6, conectadas entre si através de túneis. Os datagramas IPv6, são encapsulados nos datagramas IPv4, utilizando assim a estrutura do *backbone* IPv4 nesta etapa de transição. A tendência é que estas conexões no *backbone* deixem de ser tuneladas, havendo somente comunicação IPv6/IPv6.

Atualmente, o *6bone* conta com 33 países registrados. O Brasil, recentemente através da RNP, estabeleceu sua conexão com *6bone* . A conexão foi estabelecida através de um túnel conectando a RNP à uma empresa comercial (Cisco) dos Estados Unidos.

5 GERÊNCIA DE REDES

Uma rede de computadores normalmente é composta por uma gama de equipamentos com características peculiares, o que a torna consideravelmente complexa diante da sua diversidade. Para que estas redes, principalmente as de grande porte, operem e funcionem adequadamente será necessário tê-las sob controle. Ter uma rede sob controle significa definir uma configuração adequada a ela e a cada dispositivo que a compõe; monitorá-la constantemente para detectar e recuperar as falhas o mais breve possível, antes mesmo que seus usuários percebam, garantindo assim que seus recursos estejam sempre disponíveis quando requisitados.

Para realizar estas atividades de uma maneira eficiente, uma série de regras foram definidas a fim de estruturar o gerenciamento. A partir delas, os protocolos foram desenvolvidos e implementados [PRA95].

Existem várias instituições mundiais que têm desenvolvido serviços, protocolos e arquiteturas para o gerenciamento de redes. Atualmente as três principais são:

- ISO (*International Organization for Standardization*): desenvolveu um modelo de gerenciamento seguindo o padrão OSI (*Open System Interconnection*), tendo como protocolo de gerenciamento mais conhecido o CMIP (*Common Management Information Protocol*). O modelo OSI é o mais completo, abrangendo inúmeras funções de gerenciamento;

- ITU-T (*Telecommunication Standardization Sector T of the International Telecommunication Union*): desenvolveu o padrão TMN (*Telecommunication Management Network*) para gerenciamento de equipamentos de telecomunicações, sendo um complemento do Modelo OSI de gerenciamento;

- IETF (*Internet Engineering Task Force)*: desenvolveu o protocolo de gerenciamento padrão da *Internet*, o SNMP (*Simple Network Management Protocol*).

O SNMP vem sendo amplamente utilizado pela comunidade *Internet*. Por sua vez, o modelo OSI tem sido aplicado nas telecomunicações através do TMN. Nos outros setores de comunicação, atualmente, não existe nenhuma implementação completa deste modelo, devido principalmente a falta de recursos que o operacionalizem.

Na seqüência deste capítulo, apenas as áreas funcionais do Modelo OSI de gerenciamento e os conceitos básicos do protocolo SNMP serão apresentados, uma vez que são os temas relevantes para composição deste trabalho.

5.1 Áreas Funcionais de Gerência de Redes

No modelo de gerenciamento OSI foram definidas 5 áreas funcionais:

- gerenciamento de falhas: grupo de funções que permitem a detecção, isolamento e reparo de falhas. Dentre estas funções pode-se citar [PRA95]:
  - manter e examinar arquivos com avisos de erros (*logs* de erros);
  - receber e agir diante de notificações de erros;
  - traçar e identificar falhas;
  - realizar testes de diagnóstico;
  - corrigir falhas;

- gerenciamento de configuração: é utilizada para identificar e controlar os elementos que compõe uma rede [BLA94]. Tem como funções principais [PRA95]:
  - identificar os componentes da rede;

- manter a configuração corrente;
- guardar as mudanças na configuração;
- inicializar e finalizar os componentes das redes;
- alterar os parâmetros de rede (Por exemplo: tabelas de roteamento).

O gerenciamento de configuração identifica as mudanças significativas e modela a configuração dos recursos físicos e lógicos da rede [BRI97];

- gerenciamento de contabilização: grupo de funções que possibilita estabelecer e identificar os custos associados a cada recurso na rede. O gerenciamento de contabilização possui as seguintes funções [PRA95]:
  - informar os custos aos usuários;
  - informar as expectativas de custo;
  - definir os limites de custo (Por exemplo: desabilitar conexões de telefone);
  - combinar custos.

O gerenciamento de contabilização permite que seus usuários e gerentes definam limites na utilização e negociação de recursos adicionais [BLA94];

- gerenciamento de desempenho: As funções nesta área estão centradas na monitoração e avaliação do desempenho da rede. Essa área funcional de gerência é utilizada para determinar se um sistema está sendo utilizado com eficiência, através da avaliação de alguns fatores [BLA94]:
  - o *throughput* (escoamento);
  - tempo de resposta.

A gerência de desempenho é necessária para otimizar a qualidade de serviço oferecida pela rede;

- gerenciamento de segurança: grupo de funções que permitem manter a política de segurança adotada pela rede [BRI97] , fazendo com que seja protegida contra atitudes indevidas e acesso não autorizado.

Todas essas áreas funcionais interagem entre si, permitindo que informações obtidas por uma delas sejam fornecidas a outra área [BRI97]. Como exemplo pode-se citar a relação entre gerenciamento de desempenho e gerenciamento de configuração; os *logs* de desempenho podem ser utilizados no gerenciamento de configuração para definir eventuais mudanças necessárias na configuração.

5.2 Protocolo SNMP

No final da década de 80, a *Internet* e as redes que a compõem tiveram um crescimento considerável. As técnicas de gerência *ad hoc* (Por exemplo: ping, traceroute), não eram mais suficientemente adequadas ao gerenciamento. O SNMP surgiu desta necessidade. Desde a sua adoção pela comunidade *Internet* em 1988 [CER88], onde sua especificação e códigos foram disponibilizados, este protocolo tem sido implementado em centenas de produtos, desde os mais simples como *hubs* aos mais complexos como roteadores e *mainframes* [PRA95].

O protocolo SNMP manteve as características da arquitetura *Internet*, ou seja, simplicidade e fácil implementação. Estas características permitiram que fosse rapidamente adotado. Entretanto, devido também a sua simplicidade, ele não consegue atingir todas as necessidades de gerenciamento, principalmente das redes atuais, cada vez mais complexas. Para resolver parcialmente estas deficiências, principalmente as relacionadas com questões de segurança, uma nova versão do protocolo foi desenvolvida, o SNMPv2. Entretanto, este protocolo divergiu em duas versões V2u e V2*, impedindo a sua completa adoção. Atualmente, a versão 3 deste protocolo está sendo desenvolvida para fornecer a convergência dos conceitos e elementos técnicos do V2u e V2*. O desenvolvimento do SNMPv3 é crucial para a continuação do sucesso do SNMP [MUN98].

O gerenciamento *Internet* através de qualquer uma das versões do SNMP segue os seguintes princípios [FEI97]:

- todos os sistemas conectados à rede devem ser gerenciáveis através do SNMP;
- o custo de adicionar gerenciamento de rede ao sistemas existentes deve ser mínimo;
- deve ser relativamente simples estender as capacidades de gerenciamento dos sistemas existentes;
- o gerenciamento de redes deve ser robusto. Mesmo na presença de falhas, um grupo mínimo de funções de gerenciamento deve estar disponível.

5.3 O Modelo SNMP

Como dito anteriormente, uma rede é composta por uma diversidade de dispositivos, como computadores, impressoras, terminais, *hub*, *switches*, roteadores, entre outros. Cada um destes dispositivos possui informações de configuração, desempenho, erros, que são fundamentais na monitoração, avaliação e manutenção da rede. Na realidade tem-se uma base de dados distribuída de informações de rede.

Apesar da *Internet* não definir um modelo de gerenciamento, ela segue o padrão dos modelos de gerência que são compostos por nós gerenciáveis e suas informações, uma ou mais estações de gerenciamento e um protocolo de gerenciamento.

Um nó é dito gerenciável quando permitir que seja supervisionado e controlado por uma estação de gerenciamento. Qualquer dispositivo na rede, para se tornar um nó gerenciável, deverá ter a implementação de um **agente**. Da mesma forma, um estação será de gerenciamento se possuir a implementação de um **gerente** (ver Figura 5.1), cuja função será enviar e receber mensagens ou notificações do agente.

Figura 5. 1 Relação Agente *versus* Gerente

5.4 MIB (*Management Information Base*) *Internet*

As MIBs, ou bases de informações de gerência, são compostas pelas informações de gerenciamento e pelos objetos gerenciados. Um objeto gerenciado é definido como a unidade da informação de gerenciamento [ROS94]. A comunicação e o processamento de dados são os recursos que podem ser gerenciáveis através da utilização de um protocolo [PRA95]. A informação de gerência, por sua vez, é aquela associada com um objeto gerenciado, que é operada pelo protocolo para controlar[1] e monitorar[2] aquele objeto. As informações de gerenciamento referentes ao objetos gerenciados residem na MIB. Ela define o conteúdo da informação que é transportada através do protocolo de gerenciamento. Os protocolos de gerenciamento não operam diretamente nos objetos gerenciados, mas sim na MIB. A estrutura das MIBs possui as definições dos seguintes itens:

- os elementos que são gerenciáveis;
- de que forma esses elementos são acessados pelos usuários;
- como eles podem ser reportados.

Apesar destas diferenças conceituais e sua estrutura, tradicionalmente define-se uma MIB como um conjunto de objetos gerenciados. Estes objetos e suas instâncias[3] são representados por variáveis. Às variáveis são atribuídas definições que informam exatamente quais serão seus atributos.

[1] Controlar – atualizar as informações de gerenciamento que não estão adequadas.

[2] Monitorar – requisitar e avaliar as informações de gerenciamento.

[3] É uma ocorrência do objeto gerenciado.

5.5 Estrutura da Informação de Gerenciamento

A estrutura da informação de gerenciamento (SMI - *Strutcture of Management Information*) define as regras para a descrição da informação de gerenciamento [ROS94]. O SMI *Internet* atende às necessidades para que as variáveis que representam os objetos gerenciados da rede sejam adequadamente definidos. Sua estrutura é em forma de árvore (ver Figura 5.2), cuja função primária é definir o nomes das variáveis da MIB. Se uma nova tecnologia for adicionada ao sistema de gerenciamento, um comitê é criado e é inserido um novo nó na árvore. O SMI é definido utilizando a linguagem ASN.1 (*Abstract Syntax Notation 1*), permitindo que a MIB possa ser definida e categorizada de acordo com a estrutura hierárquica definida. A Figura 5.3 apresenta a árvore da MIB *Internet*.

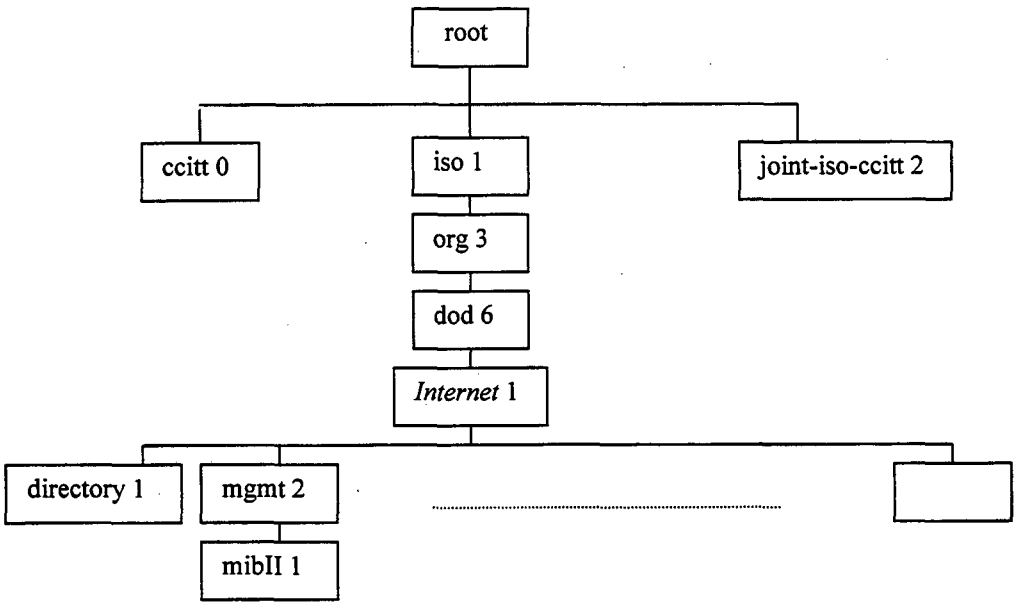


Figura 5. 2 Árvore SMI

Os objetos são representados através de uma seqüência de números inteiros não negativos, representado os nós da árvore hierárquica. Cada nó pode ter um rótulo associado, cuja função é ajudar usuários e projetistas a compreender melhor qual a função da variável.

Exemplo:

Identificador do Objeto: 1.3.6.1.2.1.1

Rótulo : iso.org.dod.*Internet*.mgmt.system.sysDescr

A MIB *Internet* foi definida originariamente em 8 grupos de objetos: *system, interfaces, address translation,* IP, ICMP, TCP, UDP, EGP. Posteriormente a MIB II foi lançada agregando mais dois grupos: *transmission* e SNMP. Cada um destes grupos define operações e novos grupos. Por exemplo: o grupo *system* descreve (1) o nome e versão do hardware, bem como o sistema operacional e o software de rede, (2) o nome do grupo na hierarquia, (3) indica quando o sistema foi reinicializado.

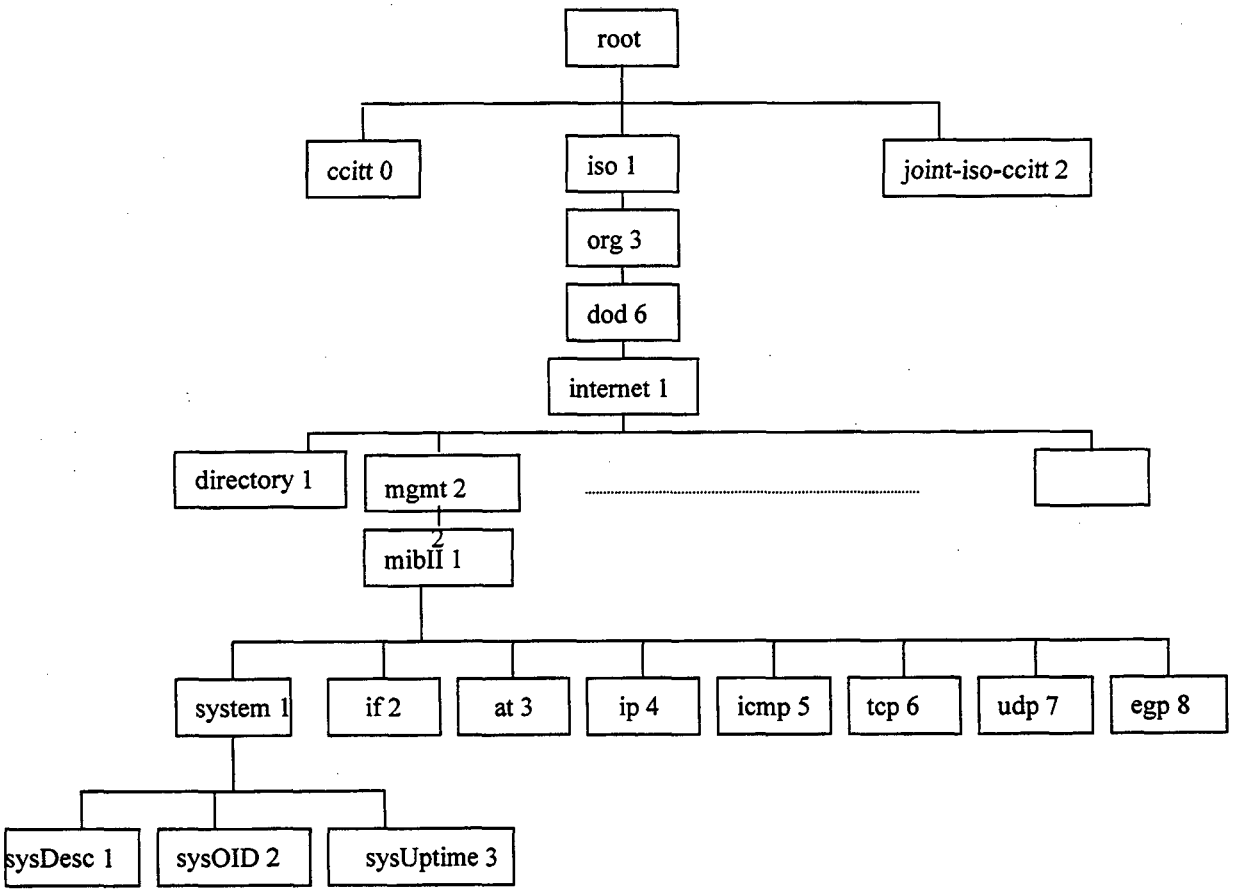


Figura 5. 3 MIB *Internet*

Os objetos são descritos através de 5 elementos (ver figura 5.4):

- descritor de objeto (*object descriptor*): descreve o objeto em texto ASCII;
- sintaxe (*syntax*): define o tipo de dado (inteiro, octeto, etc);
- definição (*definition*): descrição textual do tipo de objeto;
- acesso (*access*): define o nível de acesso às instâncias do objeto gerenciável (*read-only, write-only, read-write* ou *not-accessible*);
- estado (*status*): informa obrigatoriedade da implementação da instância do objeto. O valores possíveis são *mandatory, optional* e *obsolete*.

```
OBJECT:
        ifType {ifEntry 3}

Syntax:
        INTEGER {
                other (1)
                regular1822(2)
                .....
                    }

Definition:
        Tipo de interface ... imediatamente "abaixo" da camada de rede IP

Access:
        read-only

Status:
        mandatory
```

Figura 5. 4 Tipo de Interface (*ifType*)

5.6 Transporte

O SNMP foi projetado para operar sobre a camada de transporte UDP (ver Figura 5.5), podendo entretanto operar sobre outros protocolos.

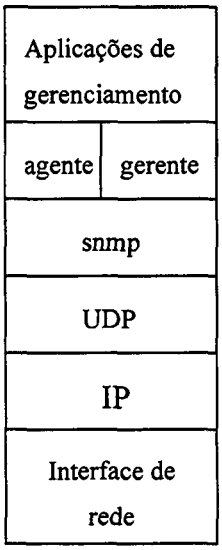


Figura 5. 5 Camadas SNMP

Os principais motivos que fizeram do UDP o protocolo escolhido foram:

- simplicidade: pode ser facilmente implementado, com pouca quantidade de código [FEI97];
- redução da carga de processamento: como a estratégia de gerenciamento adotada pelo SNMP é baseada em *pollings*, ou seja, o gerente SNMP pode ser programado para periodicamente requisitar informações de gerenciamento aos seus agentes, este tipo de estratégia, dependendo da quantidade de dispositivos de rede e da periodicidade com que os *pollings* são executados, podem causar uma considerável sobrecarga de processamento e utilização da rede. Por não ser orientado à conexão, o

processamento é muito mais leve, fazendo com que os recursos da rede sejam menos utilizados [BLA94];

- estresse da rede: a rede pode ter em certas ocasiões um alto grau de utilização. Se o protocolo fosse orientado à conexão, a própria fase de conexão seria dificultada, causando um atraso no envio do pacote de gerenciamento, prejudicando o próprio gerenciamento, exatamente no momento em que é mais necessário [ROS94];
- melhor esforço: os protocolos não orientados à conexão trabalham baseados no melhor esforço. Isso significa que não garantem que os dados alcancem seu destino. Entretanto, é dado a garantia que será feito o melhor esforço. Dessa forma, mesmo em caso de falhas, algum dado pode alcançar seu destino. O gerenciamento dessa forma pode ser possível, apesar de limitado. Por outro lado, os protocolos orientados à conexão são projetados baseados no "tudo ou nada", ou seja ou todos os dados serão entregues ou nada será entregue. Se os dados não podem ser entregues a conexão deverá ser reiniciada [PRA95].

5.7 Mensagem SNMP

Os gerentes e agentes se comunicam através da troca de mensagens SNMP. Esta mensagens são compostas por um cabeçalho padrão e uma PDU[4] (*Protocolo Data Unit*) específica (ver Figura 5.6). O cabeçalho inclui a versão do protocolo SNMP e nome da comunidade. O primeiro campo garante que agentes e gerentes estejam trocando mensagens compatíveis. O segundo campo, funciona como um senha de acesso a informações. Elas só poderão ser lidas ou escritas, baseadas no nome da comunidade. Quando um agente recebe uma solicitação do gerente, imediatamente será feita a verificação da comunidade. Se a comunidade for igual a definida pelo agente o gerente terá

[4] Termo genérico atribuído a forma como um dado de uma aplicação é empacotado para transmissão.

o acesso, caso contrário um mensagem será retornada pelo agente indicando falha na autenticação.

versão	comunidade	PDU

Figura 5. 6 Mensagem SNMP

Existem somente 5 PDUs definidas pelo SNMPv1 (ver figura 5.7):

- *GetRequest*: utilizada para requisitar um ou mais valores da MIB do sistema de gerenciamento;
- *GetNextRequest*: habilita o gerente a recuperar os valores sequencialmente;
- *SetRequest*: habilita o gerente a atualizar o conteúdo das variáveis;
- *GetResponse* (*Response* no SNMPv2): retorna o resultado das PDUs *get, get-next* e *set*;
- *Trap*: habilita um agente a reportar importantes eventos ou problemas.

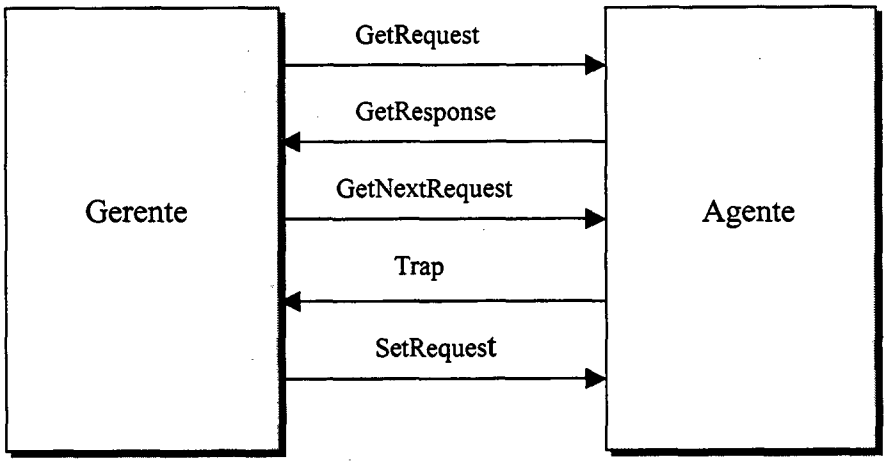


Figura 5. 7 PDUs SNMP

A comunicação entre agente e gerente se dá de maneira confirmada. O gerente toma a iniciativa através do envio das seguintes PDUs: *GetRequest,*

GetNextrequest ou *SetRequest*. Depois de receber uma destas mensagens, o agente responde com a PDU *Response*. Esta PDU transporta a informação requisitada ou indica uma possível falha na requisição.

O agente também pode iniciar a comunicação - acontece quando o agente detecta algum evento extraordinário, como a reinicialização do dispositivo ou a mudança de estado de uma das suas interfaces. Diante de tais eventos, o agente envia uma PDU *Trap* ao gerente, este por sua vez não necessita confirmar o recebimento. Não cabe ao SNMP definir o que será feito após o recebimento de um *Trap*. Esse tratamento será dado pela aplicação de gerência.

6 TECNOLOGIAS *WEB* E JAVA E FERRAMENTAS DE GERENCIAMENTO

6.1 *Web*

Existe uma quantidade grande de informações que podem ser alcançadas através da rede *Internet*. Estas informações são organizadas em centros globais de dados conhecidos como *Web*.

O sistema *Web* permite acessar um universo de informações *online* através da interação com uma simples interface de usuário. Esta interface opera sem que o usuário necessite saber onde as informações estão localizadas, como estão armazenadas, ou qual sistema é usado para gerenciá-las.

O usuário *navega* pelas informações que foram criadas por várias pessoas, em diferentes computadores, vindos de sistemas de informação e base de dados existentes. O *Web* incorpora informações vindas dos mais básicos sistemas de informação, incluindo informações hipertexto, multímidia e *Applets*.

O projeto *Web* foi iniciado por Tim-Berners-Lee e possui uma interface para os protocolos de comunicação existentes (FTP, Telnet, Gopher, NTTP, SMTP, Pop, etc.) bem como para os vários formatos de dados, sons e imagens existentes (gif, jpeg, mpeg, etc.)

O *Web* utiliza como protocolo de comunicação o HTTP (*HyperText Tranfer Protocol*) para realização da transferência das páginas *Web* e para a composição destas páginas utiliza a linguagem HTML (*HyperText Markup Language*).

6.1.1 OPERAÇÃO DO *WEB*

A visão do mundo *Web* é de documentos referindo-se a outros, através de *links*. Por sua operação de ligação pode ser comparada graficamente a uma teia de aranha, chamado de *Web* (teia). Esta visão simplificada é conhecida como paradigma hipertexto. O texto aparece na tela como um documento com partes sensitivas representado os *links*.

O *Web* divide-se em dois sistemas: o cliente e o servidor. O programa cliente é representado por interfaces conhecidas como *browsers*. Através dos *browsers* são apresentadas informações (páginas) de uma maneira eficiente e amigável oferecidas pelo servidor. A máquina cliente pode requerer uma pesquisa, digitando um endereço *Web* ou pode seguir um *link* para outro documento. Em ambos os casos, o cliente envia um pedido ao servidor, frequentemente uma máquina remota, requisitando o envio de uma nova página.

Para tornar possível a transferência de hipertextos, uma estrutura de endereçamento faz-se necessária, capacitando o WWW localizar com facilidade o documento solicitado. A estrutura adotada foi o URL (*Uniforme Resource Locator*). Ou seja, um localizador de recursos, que define o caminho de acesso a um determinado arquivo. O URL utiliza a notação apresentada na Figura 6.1. Deste modo, pode-se exemplificar o URL ***http://www.ufsc.br/arq.html***. Neste caso, o ***http*** será o protocolo utilizado, ***www.ufsc.br*** é o nome da máquina servidora e ***arq.html*** é o nome do arquivo a ser requisitado.

<protocolo utilizado>://<nome da máquina servidora>/<caminho do arquivo>

Figura 6. 1 Notação utilizada pelo URL

Além da definição da estrutura de endereçamento, é preciso definir como os documentos hipertexto fluem através da rede, isto é, o conjunto de regras que permitirá os envio e o recebimento de um documento hipertexto. A solução encontrada foi a definição de um protocolo de transferência de arquivos hipertexto chamado HTTP.

O HTTP, além de ser um protocolo de transferência hipertexto, é também utilizado para recuperar informações com a necessária eficiência para fazer saltos hipertexto. O HTTP não transfere apenas documentos HTML. Embora a compreensão HTML seja requerida por clientes *Web*, o HTTP é utilizado para recuperar documentos em vários formatos. Isto é feito pelo cliente enviando uma lista de formatos que ele pode manusear, e o servidor responde em dados em formatos que ele possa produzir. Isto permite que formatos particulares possam ser usados, sem que haja a necessidade de padronização destes. Isto é importante para usuários finais que compartilham dados em formas sofisticadas.

O HTTP permite operações de recuperação de documentos e pesquisa em textos. Este protocolo mapeia cada pedido para uma conexão TCP (*Transfer Control Protocol).* Os objetos HTTP são identificados pelo protocolo, pelo nome do servidor correspondente e pelo caminho do arquivo onde reside o conteúdo do objeto. Partes do documento podem ser também especificadas.

O HTML é uma linguagem de marcação hipertexto formada por uma uma coleção de estilos usados para definir os componentes *Web.*

Além dos conceitos apresentados, o CGI (*Common Gateway Interface*) é muito utilizado na tecnologia *Web.* Ele consiste em um padrão que permite a execução de programas executáveis e *shell scripts* no servidor. Como exemplo pode-se citar a execução de um *shell script* que gera automaticamente uma outra página *Web*, ou que faça consultas a uma base de dados. Através de um CGI será permitido a execução de qualquer programa na estação servidora. Critérios de segurança devem ser adotados para evitar que tais executáveis tenham acesso a ações indesejáveis no servidor.

6.2 Java

A principal razão para utilizar a linguagem Java no sistema de gerenciamento de redes é que ela apresenta a combinação de várias características que são peculiares à esta linguagem [VOG97]:

- portabilidade através da várias plataformas;
- programação *Internet*;
- linguagem orientada a objetos.

6.2.1 PORTABILIDADE DAS APLICAÇÕES ATRAVÉS DE DIVERSAS PLATAFORMAS

Os programas em Java são altamente portáveis devido a geração de uma representação em *byte-code* construídas pelo compiladores Java. Neste caso, o código produzido após a compilação não é específico para a plataforma. deste modo, programas escritos em Java são independentes de plataforma. Com isto a Java possui uma significante vantagem sobre as demais linguagens de programação (principalmente em aplicações clientes), uma vez que um único código (*byte-code*) será utilizado em qualquer plataforma, sem ter que haver qualquer recompilação. Consequentemente, os custos de desenvolvimento e manutenção de aplicações podem ser reduzidos.

6.2.2 PROGRAMAÇÃO *INTERNET*

O *byte-code*, gerado por um compilador Java, pode ser executado tanto por interpretadores, como por *browsers* (que simulam a implementação de uma máquina virtual baseada em *byte-code*). De fato, *browsers* com suporte a Java estão se transformando na interface de usuário gráfica universal, já inseridos no dia-a-dia dos usuários.

A linguagem Java permite a implementação de gerentes SNMP através da geração de *applets*. Essa possibilidade de gerenciamento distribuído contribui para a disseminação do gerenciamento.

6.2.4 LINGUAGEM DE PROGRAMAÇÃO ORIENTADA A OBJETOS AMIGÁVEL

Java é uma linguagem de programação orientada a objetos com características similares ao C++. Entretanto ela se mostra mais amigável, uma vez que diminui a responsabilidade do programador no gerenciamento de memória, não há o conceito de ponteiros e utiliza uma sintaxe menos confusa. Esta linguagem apresenta também características não disponíveis no C ou C++, como, por exemplo: *garbage collection* automática, gerenciamento de exceções e um suporte integrado a *threads*..

6.2.5 *APPLETS E APPLICATIONS*

A linguagem Java permite como produto final a criação de dois tipos de programas: *applets* e *applications*. Os *applets* são aplicações desenvolvidas para serem executadas no ambiente de um *browser* ou de uma ferramenta similar. Os *applets* são objetos derivados da classe *java.applet.Applet.*

Para serem acessados através de uma página *Web*, os applets são referenciados em documentos escritos em HTML. Na Figura 6.2 é apresentado um exemplo de uma página Web escrita em HTML, que referencia um *applet* Java.

```
<title>
Apenas um exemplo
</title>
<h1>
Apenas um Exemplo
</h1>
<center>
<! -- Aqui o applet é chamado -->
<applet code=ApenasUmExemploApplet.class width=400 height=80>
</applet>
</html>
```

Figura 6. 2 Exemplo de uma Página *Web* escrita em HTML com *Applets* Java

Applications, por sua vez, são aplicações destinadas a serem executadas por um interpretador. Necessariamente, uma *application* deve conter uma função *main,* através da qual ela será iniciada. Uma diferença fundamental entre *applets* e *applications* reside na questão de segurança. Como os *applets* são executados por uma máquina virtual Java, presente no *browser,* e este está instalado em uma máquina cliente, não é permitido ao *applet* Java acessar recursos desta máquina, como por exemplo arquivos locais. O objetivo desta restrição é prevenir que *applets* atuem como vírus, alterando, removendo e executando programas na máquina local. Outra restrição imposta aos *applets* está relacionada diretamente com a rede, ou seja, é permitido apenas que realize a abertura de conexões *sockets* para o *host* de onde estes *applests* foram transferidos (servidor *Web*). Deste modo evita-se que, através destes *applets,* outras máquinas presentes na rede possam ser indevidamente acessadas.

6.2.6 ANÁLISE COMPARATIVA ENTRE *APPLETS* JAVA E HTML

Por serem executados na máquina virtual Java, presente no *browser* de cliente, os *applets* diminuem consideravelmente a carga de processamento relativa a um servidor, se comparado com páginas HTML que necessitem a interação com aplicações (programas em C, *Shell scripts*, etc.), onde o processamento ocorrerá totalmente na máquina servidora. Por exemplo, o cadastro de uma pessoa no sistema de *e-mail* de uma empresa. Os dados devem ser enviados ao servidor que executará uma aplicação através de um CGI (*Common Gateway Interface*) para analisar e processar os dados. Na ocorrência de erros relativos à entrada de dados, a verificação se daria no servidor. No caso de um *applet* Java esta avaliação dos dados será feita pelo *applet*, utilizando para isso recursos da máquina cliente, enviando posteriormente os dados ao servidor para o cadastro do usuário.

6.3 Ferramentas de Gerenciamento

6.3.1 *AGENT BUILDER*

A *Advent Agent Builder* é uma ferramenta gráfica desenvolvida pela *Advent Network Management Inc.*, implementada em Java que permite ao usuário definir, criar e modificar MIBs SNMP. Esta ferramenta permite também a implementação de agentes SNMP em Java.

A *Agent Builder* possui três módulos integrados:

- *SNMP MIB Builder*: utilizado para criar e modificar MIBs SNMP. Seu ambiente oferece, de uma maneira gráfica, os detalhes da sintaxe ASN.1 adotados na definição da estrutura MIB SNMP;
- *Extensible SNMP Agent*: é utilizado para instrumentar MIBs SNMP e servir como um agente SNMP baseado em *java*, permitindo que gerentes e aplicações SNMP o acessem.

Através deste ambiente a MIB e o agente que estão sendo construídos podem ser avaliados e testados, através de simulações. Um agente é inicializado e aplicações gerentes podem acessá-lo, verificando se as respostas e os procedimentos estão corretos. Esta porção do *Agent Builder* é a que será instalada e inicializada nos sistemas que executarão o agente SNMP.

- SNMP MIB *Browser*: este módulo permite que uma MIB seja avaliada. Neste caso o usuário pode checar os módulos da MIB que está sendo definida ou acessar o agente em desenvolvimento, sem ter que interromper o processo de prototipação e criação.

Por ser baseada em Java, o agente desenvolvido apresenta os benefícios desta linguagem, permitindo que seja executado nas mais diversas plataformas. Deste modo é possível a criação de uma rica e ágil infra-estrutura de gerenciamento de redes.

6.3.2 *NETMONITOR*

A *Advent NetMonitor* também é uma ferramenta gráfica desenvolvida pela *Advent Network Management Inc.*, que pode ser utilizada para construir *applets* Java capazes de monitorar e controlar dispositivos gerenciáveis através do protocolo SNMP.

Esta ferramenta possui a mesma funcionalidade que outras aplicações utilizadas para a construção de ambientes gráficos. A principal diferença está no fato de que os objetos gerados terão embutidas operações SNMP. A *NetMonitor* cria um código fonte em Java. Este código, após ser compilado, pode ser testado através do *Applet Viewer* ou de um *browser* habilitado para Java.

Os *applets criados pela NetMonitor podem ser executados* em um *browser* e são capazes de realizar a comunicação com dispositivos de rede habilitados para o gerenciamento SNMP.

A *NetMonitor* apresenta as seguintes características:

- Aplicação de gerenciamento é independente de plataforma, uma vez que *applets* podem ser executados em qualquer plataforma que possua um *browser* instalado;
- o ambiente de gerenciamento pode ser construído, baseado nas reais necessidades do gerente de redes;
- este ambiente permite que os agentes SNMP sejam facilmente testados.

6.3.3 *ADVENT* SNMP API

As duas ferramentas apresentadas anteriormente utilizam um conjunto de classes chamadas de *Advent SNMP API* (*Application Program Interface*) que permitem o envio e recebimento de primitivas SNMP. Esta classes podem ser utilizadas por qualquer programador Java, para desenvolver suas próprias aplicações, de gerência ou para incrementar a implementação de gerentes, com algumas características ainda não disponibilizadas pela *NetMonitor*.

7 AUTOCONFIGURAÇÃO

Neste capítulo serão apresentados os principais protocolos e conceitos envolvidos no processo de autoconfiguração de nós *Internet* que adotam o protocolo IPv6.

7.1 IPv6 *Neighbor Discovery Protocol* (NDP)

O protocolo IPv6 *Neighbor Discovery* possui a funcionalidade do protocolo ARP (ver Seção 3.3.2), ICMP *Router Discovery Protocol* e da mensagem ICMP de redirecionamento (ver Seção 3.5) presentes no protocolo IPv4.

Este protocolo define mecanismos para resolver os seguintes problemas [NAR98]:

- descoberta de roteador: possibilita que um *host* descubra os roteadores presentes no *link* ao qual ele está diretamente conectado;
- descoberta de prefixo: possibilita que um *host* descubra o grupo de prefixos de endereços atribuídos ao seu *link*, permitindo que o *host* diferencie nós que residem no seu *link*, daqueles alcançáveis somente através de roteadores;
- descoberta de parâmetros: permite que um nó aprenda parâmetros de configuração referentes a seu *link*, como, limite de *hop*, MTU do *link*, entre outros;
- autoconfiguração de endereços: oferece condições aos nós de automaticamente atribuírem endereços às suas interfaces;

- resolução de endereço: permite aos nós determinarem o endereço físico de um nó vizinho por onde o pacote passará ou será entregue, quando é dado apenas o IP destino;
- determinação do próximo-*hop* (*next-hop*): fornece o algoritmo para mapear um endereço IP destino em endereço IP de uma máquina vizinha para o qual o pacote deverá ser enviado. O próximo *hop* pode ser um roteador ou o próprio destino;
- detecção da inalcançabilidade de um nó vizinho: possibilita aos nós determinarem que um vizinho não é mais alcançável. Para um nó vizinho agindo como roteador, um roteador *default* alternativo deverá ser optado, quando se detectar a sua inalcançabilidade;
- detecção de endereço duplicado: permite a um nó determinar se o endereço que ele deseja atribuir à sua interface não está já sendo utilizado por outro nó;
- redirecionamento: possibilita que um roteador informe a um *host* qual o melhor nó para onde ele deverá encaminhar os pacotes afim de alcançar um determinado destino. Possivelmente, isto acontecerá por uma ausência no conteúdo da tabela de roteamento do *host* origem.

O protocolo IPv6 *Neighbor Discovery* se fundamenta através do protocolo ICMPv6. Ele define novos tipos de mensagens ICMPv6 através do qual sua atividade é realizada.

Afim de viabilizar os recursos citados anteriormente, o IPv6 *Neighbor Discovery* define 5 tipos de mensagens ICMPv6:

- Solicitação de Roteador (*Router Solicitation*): quando uma interface é habilitada em um *host*, este pode enviar uma mensagem *Router Solicitation* aos roteadores requisitando o envio de *Router Advertisements*. Este tipo de mensagem é utilizada para agilizar o processo de autoconfiguração de um nó, uma vez que mensagens *Router Advertisement* (ver próximo item) são enviadas periodicamente pelos roteadores e neste caso o envio destas

mensagens pode não coincidir com a necessidade de um nó em recebê-las. A mensagem *Router Advertisement* é enviada através de um endereço *multicast* ***All-routers Multicast Address*** **(FF02::2)**, permitindo assim que seja alcançada por todos os roteadores.

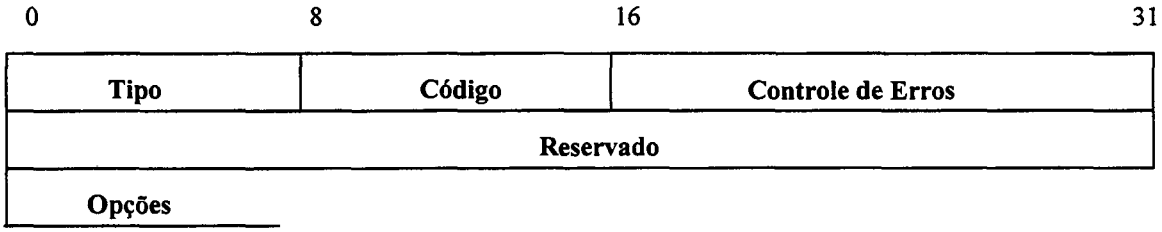


Figura 7. 1 Formato da Mensagem *Router Solicitation*

A Figura 7.1 apresenta o formato da mensagem *Router Solicitation* onde:

Tipo: possui valor 133;
Código: possui valor 0;
Controle de Erros: controle de erros do ICMPv6 ;
Reservado: este campo ainda não tem utilização, ele deve ser inicializado com zero pelo emissor , sendo posteriormente ignorado pelo receptor;
Opções: ver Seção 7.1.1.

- Aviso de Roteador (*Router Advertisement*): através desta mensagem os roteadores de um *link* informam suas presenças, bem como enviam parâmetros de configuração que devem ser adotados pelo(s) *host(s)*. Este tipo de mensagem será enviada de tempos em tempos pelos roteadores para que novos *hosts* detectem sua presença ou para que novos padrões de configuração sejam repassados. Ela será enviada para todos os nós do *link* através de um endereço *multicast* ***All-nodes Multicast Address*** **(FF02::1)**, ou então utilizará o endereço *unicast* de um determinado nó em resposta a uma mensagem *Router Solicitation.*

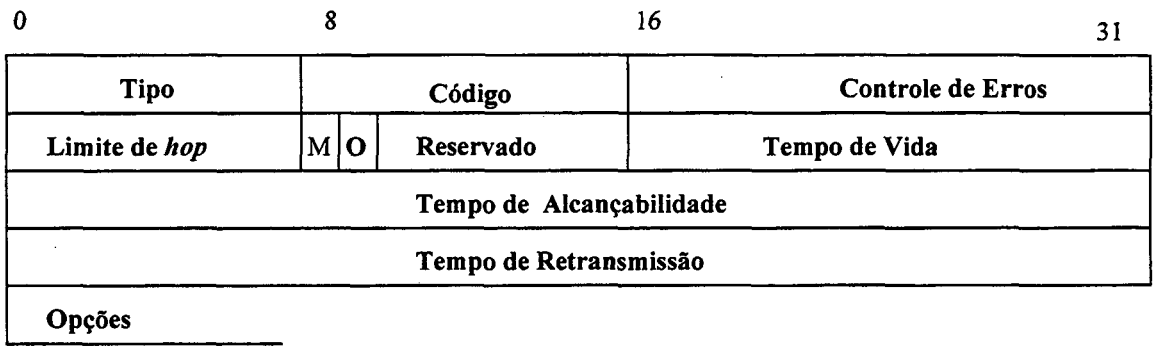


Figura 7. 2 Formato da Mensagem *Router Advertisement*

A Figura 7.2 apresenta o formato da mensagem *Router Advertisement* onde:

Tipo: possui valor 134;

Código: possui valor 0;

Controle de Erros: controle de erros do ICMPv6;

Limite de *hop*: valor *default* que deverá ser colocado no campo limite de *hop* do cabeçalho IP dos pacotes que serão enviados;

M: *flag* de "configuração" de endereço gerenciável (*Managed Address Configuration*). Quando seu valor for 1, os *hosts* deverão utilizar autoconfiguração *stateful* para obter seu endereço IP;

O: *flag* de "outra configuração *stateful*" (*Other Stateful Configuration*). Indica que os *hosts* devem utilizar autoconfiguração *stateful* para obter informações de configuração adicional (excluindo endereços);

Reservado: este campo ainda não tem utilização. Ele deve ser inicializado com zero pelo emissor , sendo posteriormente ignorado pelo receptor;

Tempo de Vida: indica por quanto tempo o roteador deverá ser mantido como roteador *default* pelos *hosts*. Um valor 0 neste campo, indica que o roteador não deverá ser adotado como roteador *default* (em segundos);

Tempo de Alcançabilidade: define o tempo em que um nó deve considerar seu vizinho alcançável depois de receber uma confirmação de alcançabilidade (em milisegundos);

Tempo de Retransmissão: define o tempo entre mensagens *Neighbor Solicitation* retransmitidas;

Opções: ver Seção 7.1.1.

- Solicitação de Vizinho (*Neighbor Solicitation*): este tipo de mensagem é enviada por um nó com o objetivo de determinar o endereço físico de um nó vizinho. Além disso permite verificar se um vizinho ainda é alcançável, baseado em uma tabela *cache* que contém o endereço físico dos nós vizinhos. Para alcançar estes nós, será utilizado um endereço *multicast* ***All-nodes Multicast Address*** **(FF02::1)**. É através deste tipo de mensagem que se descobre a duplicação de endereços.

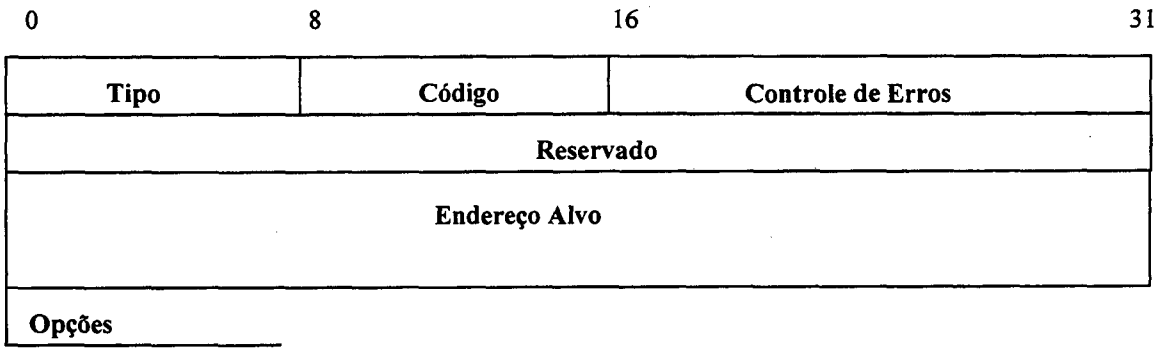


Figura 7. 3 Formato da Mensagem *Neighbor Solicitation*

A Figura 7.3 apresenta o formato da mensagem *Neighbor Solicitation* onde:

Tipo: possui valor 135;

Código: possui valor 0;

Controle de Erros: controle de erros do ICMPv6;

Endereço Alvo: endereço do alvo de uma *Solicit*ação

Opções: poderá conter o endereço físico do nó emissário. Veja Seção 7.1.1.

- Aviso de Vizinho (*Neighbor Advertisement*): esta mensagem será enviada em resposta a uma mensagem *Neighbor Solicitation*, utilizando para isso o endereço *unicast* do nó requisitante. Outra possibilidade de sua utilização é

para o envio de um aviso que anuncie a mudança de um endereço físico (ex: substituição de um placa de rede). Neste caso será utilizado o endereço *multicast* ***All-nodes Multicast Address*** **(FF02::1).**

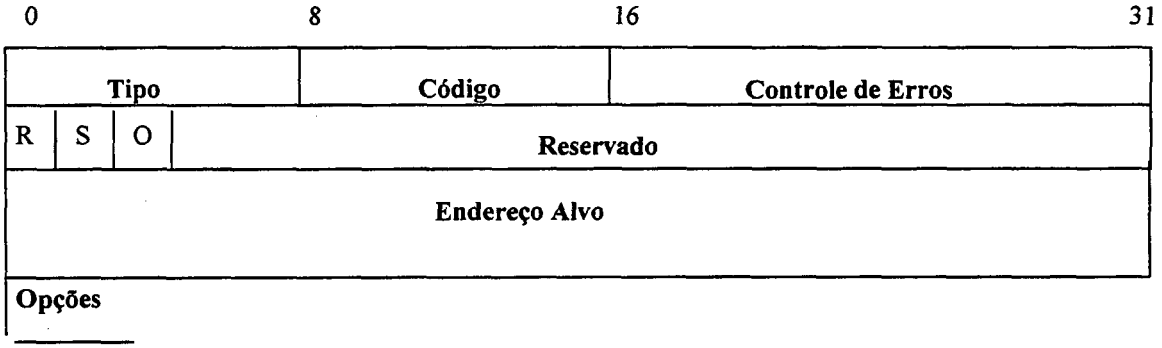


Figura 7. 4 Formato da Mensagem *Neighbor Advertisement*

A Figura 7.4 apresenta o formato da mensagem *Neighbor Advertisement* onde :

Tipo: possui valor 136;

Código: possui valor 0;

R: quando habilitado, indica que o emissário é um roteador;

S: *solicit flag*, quando habilitado indica que o aviso foi enviado em resposta a uma mensagem *Neighbor Solicitation*;

O: *override flag*, quando habilitado indica que uma substituição em uma entrada na tabela de vizinhos e uma atualização na tabela de endereços físicos;

Reservado: este campo ainda não tem utilização. Ele deve ser inicializado com zero pelo emissor, sendo posteriormente ignorado pelo receptor;

Controle de Erros: controle de erros do ICMPv6;

Endereço Alvo: para avisos solicitados, contém o mesmo endereço presente no campo de mesmo nome da mensagem *Solicit*. Para avisos não Solicitados, representará o endereço do *host* que teve seu endereço físico modificado;

Opções: endereço físico do emissário do aviso. Veja Seção 7.1.1.

- Redirecionamento (*Redirect*): este tipo de mensagem é utilizada por um roteador, para informar a um *host* qual é o melhor primeiro *hop* no caminho definido para encontrar um determinado destino. Equivalente a mensagem de redirecionamento do ICMPv4.

7.1.1 Opções

As mensagens *Neighbor Discovery* podem conter zero ou mais opções. As opções definidas são:

- Endereço Físico Fonte/Alvo: que contém como conteúdo o endereço físico de um determinado nó emissário ou destino. A opção endereço físico fonte será utilizada pelas mensagens *Neighbor Solicitation*, *Router Solicitation* e *Router Advertisement*;

- Informação do Prefixo: fornece o prefixo de rede. Normalmente aparece na mensagem *Router Advertisement;*

- Cabeçalho de Redirecionamento: este tipo de opção será utilizada nas mensagens *Redirect* e contém o pacote a ser redirecionado;

- MTU: deve ser utilizada na mensagem *Router Advertisement*, o objetivo é assegurar que todos os nós do *link* possuam o mesmo valor de MTU.

7.1.2 Estrutura de Dados

Cada *host* pertencente a um *link* manterá uma estrutura de dados (associada a cada interface de rede) para agilizar a interação com os nós vizinhos. Este dados são atualizados constantemente com informações adquiridas a partir das mensagens

Neighbor Discovery. Esta estrutura é composta por 2 tabelas e 2 listas apresentadas a seguir:

- Tabela de Vizinhos: esta tabela conterá um grupo de entradas com IP *unicast* dos nós vizinhos e de seus equivalentes endereços físicos (conexões recentes). Além disto, cada entrada contará com outros registro indicativos, como por exemplo: se o nó é um roteador ou *host*, se o nó é alcançável ou não;

- Tabela de Destinos: este tipo de tabela fará o mapeamento entre endereços IPs destino com o *next-hop* utilizado. Esta tabela é atualizada através de informações aprendidas com recebimento de mensagens *Redirect*. A tabela será composta por dois campos principais representados na Figura 7.5;

<IP destino> <Endereço IP do *next-hop* >

Figura 7.5 Campos da Tabela de Destinos

- Lista de Prefixos: esta lista conterá o grupo de prefixos de redes atribuídos ao *link* a qual o *host* está diretamente conectado. Esta lista é criada e atualizada através de informações aprendidas com recebimento de mensagens *Router Advertisements*;

- Lista de Roteadores *Default*: Esta lista conterá a relação de roteadores para os quais o *host* poderá enviar seus pacotes.

A Figura 7.6 apresenta o resultado da execução de um comando que mostra o conteúdo da tabela de vizinhos de um determinado *host*, utilizado nos experimentos práticos realizados neste trabalho.

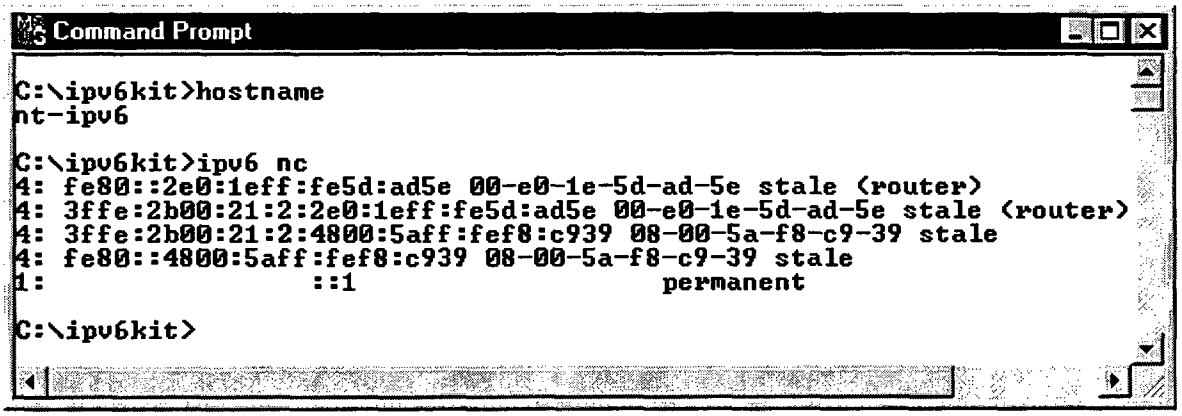


Figura 7. 6 Exemplo de Tabela de Vizinhos

7.1.3 Resolução de Endereços

Para emitir um pacote para um nó destino, o nó origem se valerá das informações contidas na sua estrutura de dados (ver Seção 7.1.2). Sempre que for necessário enviar um pacote, o nó origem checará sua tabela de destinos. Se já houver um registro, ele repassará o pacote para o *next-hop* indicado. Caso não haja um registro, o pacote será enviado para um roteador selecionado a partir da tabela de roteadores. Uma vez determinado o *next-hop*, será verificado o seu endereço físico correspondente na tabela de vizinhos. Se o endereço for encontrado o pacote será repassado. Caso contrário, o *host* deverá enviar uma mensagem *Neighbor Discovery* afim de descobrir o endereço do *next-hop* para onde o pacote será repassado posteriormente. Ao descobrir o vizinho, a tabela que mapea endereço físico para endereço IP será atualizada.

7.2 Autoconfiguração *Stateless*

A autoconfiguração no IPv6 se processa em duas etapas distintas, conhecidas como autoconfiguração *stateless* e autoconfiguração *stateful*. Na primeira etapa acontece a configuração *stateless* e na segunda a *stateful*. Entretanto, para que um *host* tenha acesso ao seu *link* e a rede global, basta apenas que a primeira etapa seja realizada.

Este capítulo apresentará as características e os procedimentos da autoconfiguração *stateless*.

7.2.1 Características

As principais características e objetivos da configuração *stateless* são os seguintes [THO98]:

- não requer configuração manual de *hosts*: não haverá a necessidade de intervenção manual por parte do administrador de rede na configuração do *host*;

- configuração mínima (se houver) de roteadores: este tipo de configuração está preocupada em oferecer as condições mínimas para que um nó tenha possibilidade de estabelecer comunicação com seus vizinhos, bem como com a rede *Internet* em geral;

- não existe a necessidade de servidores adicionais: para este tipo de configuração, apenas a presença de um roteador se fará necessária, se houver a necessidade de comunicação com nós externos ao *link*, em que o *host* em questão estiver diretamente conectado. Sem a presença de roteadores, um *host* poderá apenas gerar seu endereço *link-local*;

- permite a um *host* gerar seu próprio endereço utilizando uma combinação de informações presentes localmente e providas por roteadores: roteadores fornecem o prefixo que identifica a subrede(s) associadas ao *link* em questão, enquanto que *hosts* geram o identificador de interface.

7.2.2 Endereços *Link-Local*

O endereço *link-local* é um tipo de especial de endereço *unicast* (ver seção 4.5). O propósito deste endereço é permitir a comunicação direta de nós vizinhos através do protocolo IP. Pacotes enviados a estes endereços provenientes de outras redes nunca serão repassados pelos roteadores, sendo portanto inalcançáveis a partir de outros *links*.

Um endereço *link-local* é composto por um prefixo FE80 (16 bits), 48 bits zeros, e um identificador de interface (64 bits) e sua notação é da forma apresentada na Figura 7.7.

FE80:0:0:0:XXXX:XXXX:XXXX:XXXX

Figura 7.7 Notação do Endereço *Link-Local*

Na notação apresentada na Figura 7.7, o conjunto de letras **X** devem ser substituídas pelo endereço físico da interface no formato IEEE EUI-64, identificador global de 64 bits [IEE97]. Este formato é uma evolução do IEEE EUI-48 que adotava apenas 48 bits. Como a maioria das interfaces ainda utilizam o formato antigo, é permitido, a partir deste endereço, compor um endereço EUI-64. A regra para a formação é bastante simples, basta apenas adicionar dois octetos com o valor FF-FE entre o terceiro e quarto octeto do endereço antigo. A Figura 7.8 mostra um exemplo desta composição. O quadro superior representa um endereço de 48 bits e o quadro inferior representa um endereço EUI-64 derivado do EUI-48.

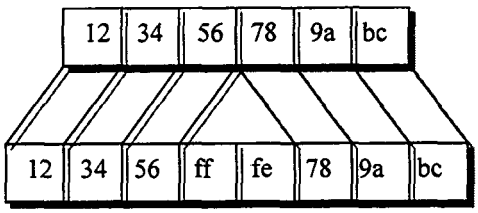


Figura 7. 8 Composição de EUI-64 a partir de um EUI-48

Para distinguir um identificador global EUI-64 original (endereços únicos alocados regularmente) de um derivado (a derivação pode gerar um endereço igual a um regularmente alocado), foi definido pelo IEEE que o sétimo bit mais à esquerda terá valor 1 para endereços reservados e valor 0 (zero) para endereços de 48 bits encapsulados em endereços de 64 bits, sendo este bit conhecido como bit "u". Entretanto, na composição de um identificador de interface para endereços IPv6 esta regra causa um pequeno problema porque 0:0:0:0 é um EUI-64 válido, que pode colidir com endereços especiais IPv6. Para evitar tais colisões, foi decidido que o bit "u" seria invertido nos endereços IPv6 que são compostos com identificador de interface. Então, um identificador de interface composto por um EUI-64 reservado deve ter o bit "u" invertido para 0 e para 1 quando utilizar um EUI-64 derivado de um EUI-48. A Figura 7.9 apresenta um exemplo dos passos envolvidos na composição de um identificador de interface a partir de um EUI-48. O valores são apresentados utilizando as notações binárias e hexadecimal.

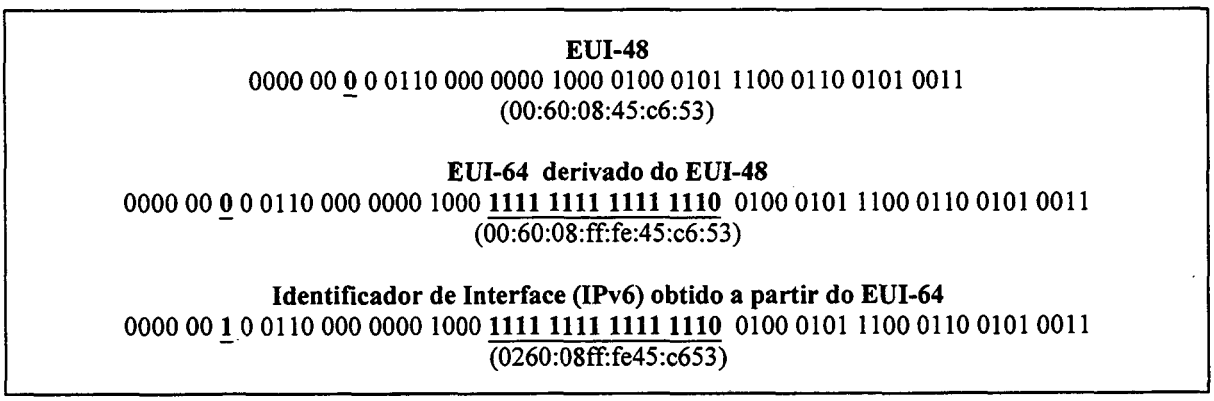


Figura 7.9 EUI-48/ EUI-64 derivado/ Identificador de Interface

7.2.3 Processo de Autoconfiguração *Stateless*

Durante a inicialização do sistema, os nós envolvidos, ou seja, *hosts* e roteadores começam o processo de autoconfiguração. Às suas interfaces é atribuído um endereço *link-local*. A atribuição deste endereço à interface só será efetivada após a

verificação da não existência de duplicidade deste endereço, isto é, a garantia de que este endereço não esteja já sendo utilizado por outro nó no *link*. O processo de verificação ocorre através do envio de mensagens *Neighbor Solicitation* com destino ao endereço que será possivelmente atribuído. Se alguma máquina responder a esta solicitação significa que este endereço já está sendo utilizado por outro nó, cabendo ao administrador de rede tomar as devidas providências para uma configuração alternativa (manual), uma vez que o processo de autoconfiguração automaticamente finaliza. O administrador deverá definir um novo identificador de interface, que não colida com um já existente. Por outro lado, se não houver resposta à solicitação o endereço *link-local* definido, este endereço é atribuído à interface.

Com a definição e atribuição do endereço *link-local,* já é possível que o nó se comunique com os outros nós que compartilham o mesmo *link* (nós vizinhos).

A próxima etapa no processo de autoconfiguração *stateless* necessita da participação de um roteador. É através desta fase que será possibilitado aos *hosts* o acesso à rede global. Se houver(em) roteador(es) presente(s) neste *link*, eles enviarão de tempos em tempos *Router Advertisements* através do endereço ***All-nodes Multicast Address* (FF02::1)**, que especificarão que tipo de autoconfiguração que um *host* deverá ter (*stateless* ou *stateful*). A não presença de roteadores indica a necessidade de invocação de uma configuração *stateful* (se houver a obrigatoriedade de acesso à rede global).

Como dito anteriormente, os roteadores do *link* em questão enviarão de tempos em tempos *Router Advertisements*. Entretanto, o envio deste aviso poderá não coincidir com a necessidade do *host.* Deste modo, ele terá que aguardar um tempo muito "longo" para receber as instruções do roteador de como proceder a autoconfiguração. Dessa forma, é facultado ao *host* enviar *Router Solicitations* para todos os roteadores, utilizando para isso, o endereço ***All-routers Multicast Address* (FF02::2)**. Os roteadores que receberem esta *Solicit*ação responderão com *Router Advertisements*.

Os *Router Advertisements* contém dois *flags* indicando que tipo de configuração *stateful* o *host* deverá adotar (no caso de haver este tipo de configuração). A configuração *stateful* poderá ser do tipo configuração do endereço gerenciável (*managed address configuration*) ou outra configuração *stateful* (*other stateful configuration*). Respectivamente, a primeira irá indicar que o endereço, além de outras configurações, será

obtido através do método *stateful*, e a segunda indicará que o endereço global será obtido através do método *stateless*, sendo que outras configurações (exceto endereço) poderão ser obtidas através do método *stateful*. Um fator importante a ser notado, é que o *host* poderá obter endereços através dos dois tipos de autoconfiguração simultaneamente, não sendo estes métodos mutuamente exclusivos. Sendo assim, o *host* poderá ter um endereço *link-local* e outro global obtido através da autoconfiguração *stateless*, e pelo menos mais um endereço global obtido através da autoconfiguração *stateful*, ficando claro que este mesmo *host* poderá conter outros endereços atribuídos manualmente pelo administrador da rede.

Todo este processo será repetido tantas quantas forem as vezes que os roteadores enviarem seus *Router Advertisements*. Toda vez que uma mensagem *Router Advertisement* for recebida, irá acontecer uma reavaliação da configuração do *host*, podendo haver mudanças ou não, dependendo exclusivamente do tipo de informação que chegar, proveniente do roteador.

É importante salientar que para *hosts multi-homed* a autoconfiguração ocorre independentemente em cada interface.

A Figura 7.10 apresenta a configuração de uma interface que sofreu autoconfiguração *stateless*. Neste caso, o *host* atribuiu o endereço *link-local* **fe80::**260:8ff:fe45:c653 e o endereço global **3ffe:2b00:21:2:** 260:8ff:fe45:c653. Deve-se observar que o identificador de interface é o mesmo, pois é obtido a partir do endereço físico da interface (ver Seção 7.2.2).

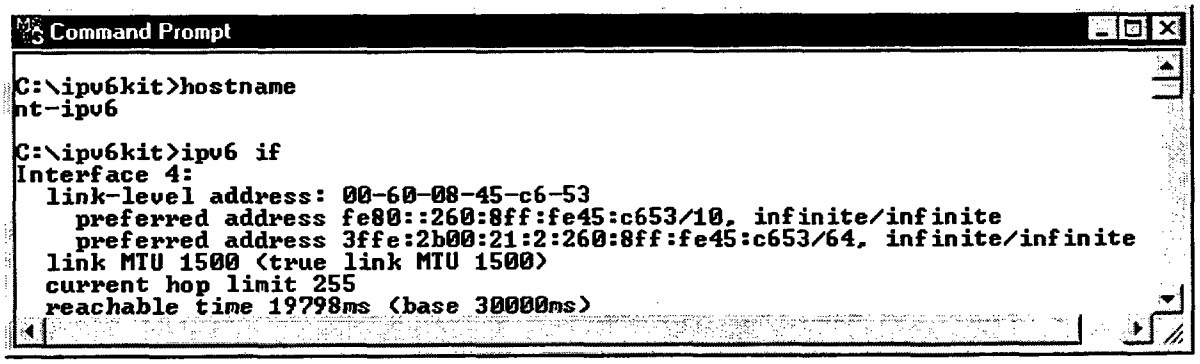


Figura 7. 10 Interface Autoconfigurada

7.2.4 Tempo de Vida do Endereço

Os endereços IPv6 atribuídos às interfaces terão um tempo de vida fixado (podendo ser infinito). Para cada endereço atribuído estará associado um tempo de vida. A partir do momento que este tempo se esgotar, haverá a possibilidade que ele seja delegado a uma nova interface de um outro nó. Para evitar maiores problemas, uma vez que este tempo pode se esgotar no meio de uma conexão, foi adotado um método para tratar este impasse. O tempo de vida foi dividido em duas fases: *preferred* e *deprecate*. Enquanto ele estiver na fase *preferred* as comunicações envolvendo este IP serão completamente normais. A partir do momento em que o tempo de vida entrar na fase *deprecate*, a utilização deste IP é desencorajada, mas não proibida. Nesta etapa, o endereço só será utilizado por aquelas aplicações cuja mudança no endereçamento provocará a quebra da conexão. Atualmente, as aplicações que utilizam o protocolo TCP da camada de transporte necessitam que o endereço se mantenha o mesmo, durante o período de tempo que a conexão estiver aberta. Estas fases no tempo de vida de um endereço são adotadas nos dois tipos de autoconfiguração.

7.2.5 Mudança de Endereçamento de uma Rede

Devido as duas fases (*preferred* e *deprecate*) de tempo de vida de um endereço adotado pelo IPv6, a mudança de endereçamento de uma rede fica bastante facilitada. Quando a fase *deprecate* de um endereço estiver sendo alcançada, é enviado às camadas superiores um aviso indicando que este endereço está perdendo sua validade. Deste modo, as camadas superiores podem optar por um endereço com tempo de vida *preferred*. Toda vez que se desejar fazer a mudança de endereçamento de uma rede, bastará reconfigurar o roteador que enviará *Router Advertisements* contendo esta nova configuração. Como a interface de um *host* poderá conter mais que um endereço, ela se manterá com a configuração antiga até que o tempo de vida se esgote e o novo endereço será atribuído instantaneamente. Este novo endereço será adotado pelas camada superiores, tão logo o endereço antigo se torne inválido. Apesar de não ser adotado instantaneamente

pelas camada superiores, qualquer nó na rede que quiser alcançá-lo terá êxito, uma vez que este novo endereço está correntemente atribuído e válido para a sua utilização em qualquer conexão. Com isso, temos a utilização simultânea destes grupos de endereçamento.

7.2.4 Aspectos de Segurança

Os aspectos de segurança adotados na autoconfiguração *stateless* se limitam aos recursos oferecidos pelo IPv6. Se o cabeçalhos de extensão de segurança não forem adotados, esta técnica não utilizará qualquer outro procedimento. É permitido a qualquer *host* que tenha acesso físico ao *link*, que se configure e inicialize a comunicação com outros nós, sem que necessite se registrar ou se autenticar.

7.3 Autoconfiguração *Stateful* - DHCPv6

A autoconfiguração *stateless* apresenta muitas vantagens, a principal delas reside no fato de não necessitar a presença de qualquer máquina servidora além do roteador da rede. Entretanto, este método de autoconfiguração apenas se preocupa com endereçamento. Outros aspectos relevantes na configuração de um nó não são abordados. Além disso, ela apresenta inconvenientes: a ineficiência na utilização do uso do espaço de endereçamento, pois fixa o prefixo de rede em 64 bits. Neste caso o identificador de interface é baseado no identificador global EUI-64 e não tem nenhum método de controle de acesso à rede.

Para complementar o método *stateless* de autoconfiguração, tem-se o método *stateful.* Este método é responsável por oferecer todos os parâmetros de configuração básicos necessários aos nós de uma rede. Este método também permite que apenas os nós com autorização acessem o ambiente de rede envolvido.

7.3.1 Características

As principais características da configuração *stateful* são as seguintes [THO98]:

- os *hosts* obtém endereço de interface e/ou informações de configuração a partir de um servidor;
- os servidores mantém uma base de dados que armazena a relação de endereços atribuídos, bem como para que *hosts* eles foram designados;
- é utilizada quando se necessitar um controle maior dos endereços atribuídos.

O protocolo de configuração *stateful,* utilizado no IPv6, é o DHCPv6. Ele é projetado para operar segundo o modelo cliente/servidor. Deste modo os servidores contemplam seus clientes com informações de configuração ou parâmetros de inicialização para que possam integrar à rede sem conhecimento prévio de nenhuma configuração de rede além de um endereçamento inicial. A relação entre cliente e servidor acontece através da troca de mensagens, que terão principalmente a função de transmitir os dados de configuração ao cliente.

Não há a necessidade da presença de um servidor em cada *link.* Aquelas redes que não possuírem um servidor DHCPv6, podem ter como entidade intermediária um agente DHCPv6 *Relay,* que é responsável por repassar as mensagens entre clientes e servidores.

O DHCPv6 utiliza como protocolo de transporte o UDP. Apesar do UDP não ser um protocolo confiável, o DHCPv6 utiliza um sistema de retransmissões para garantir a confiabilidade entre clientes e servidores. As portas UDP utilizadas são as de número 546 e 547. O servidor ou agente[1] *Relay,* se comunicará com os clientes DHCPv6 utilizando como a porta destino 546, este por sua vez transmite mensagens para os agentes DHCP utilizando a porta 547 como destino. Um servidor DHCP deve transmitir suas mensagens para um agente *Relay* através da porta 546. A Figura 7.11 ilustra as portas

[1] Nesta seção o termo agente referencia servidor e *Relay*

utilizadas durante a troca de mensagens entre servidor/cliente, servidor/*Relay*, cliente/*Relay*.

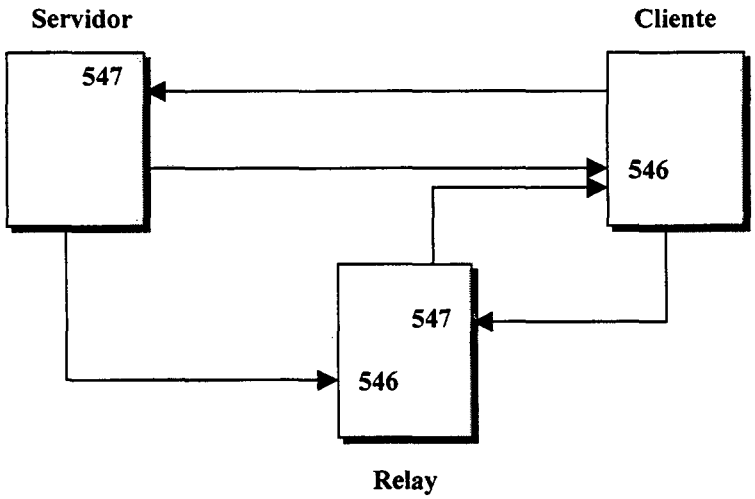


Figura 7. 11 Portas utilizadas durante a troca de Mensagens DHCPv6

7.3.1 Processo de Autoconfiguração *Stateful*

O processo de autoconfiguração DHCPv6 é realizado através de mensagens trocadas entre servidores e clientes. A Figura 7.12 apresenta a seqüência dos passos envolvidos que serão apresentados a seguir.

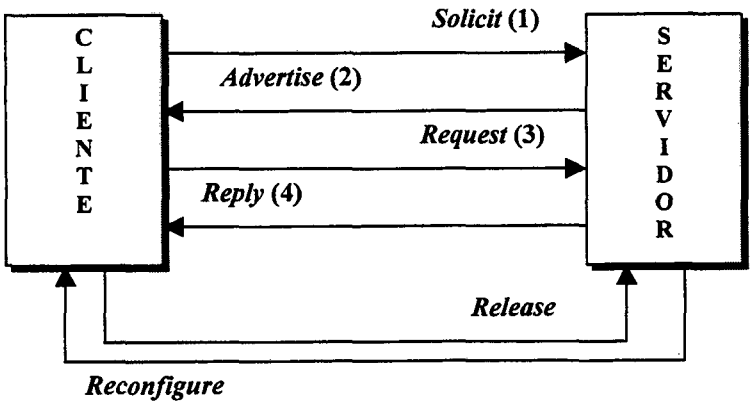


Figura 7. 12 Mensagens trocadas entre Clientes e Servidores DHCPv6

A autoconfiguração *stateful* inicializa-se através do envio de uma mensagem DHCP *Solicit* originada por uma máquina cliente. Esta mensagem é destinada para todos os agentes presentes no *link*, e possui como endereço destino, o endereço *multicast* **ALL-DHCP-Agents *Multicast* Address (FF02::1:2).**

Utilizando esta mensagem, o cliente pode localizar os agentes DHCPv6. Como endereço IP fonte, o cliente utiliza seu endereço *link-local*. Depois de montada a mensagem, o cliente a envia através da interface a ser configurada. Se eventualmente não existirem servidores DHCPv6 diretamente conectados ao mesmo *link* do cliente, possivelmente haverá um agente *Relay* que se encarregará de repassar esta mensagem a um servidor. A mensagem *Solicit* terá o preenchimento dos campos **tamanho do prefixo** e **endereço do agente *Relay*** preenchidos com seus devidos valores. O agente *Relay* repassará esta mensagem para o servidores DHCP utilizando o endereço *multicast* **ALL-DHCP-Servers *Multicast* Address (FF05::1:3).**

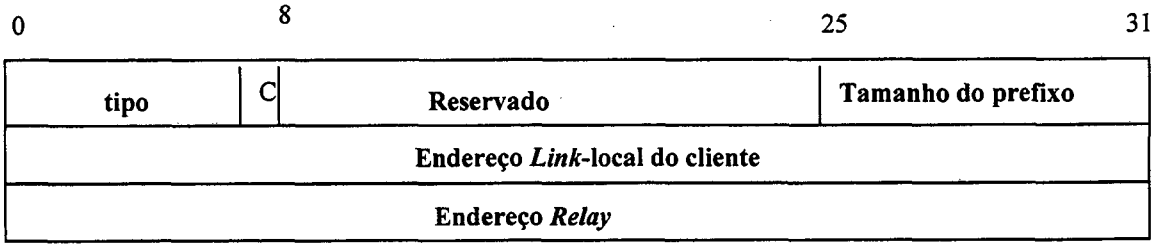


Figura 7. 13 Formato da Mensagem DHCPv6 *Solicit*

A Figura 7.13 ilustra o formato da mensagem DHCPv6 onde:

Tipo: possui valor 1;

C: se apresentar valor 1, significa que todos os servidores que receberem a mensagem devem liberar os recursos previamente associados ao cliente;

Reservado: possui valor 0;

Tamanho do Prefixo: tamanho do prefixo de rede do agente *Relay*. Este campo será utilizado quando a mensagem enviada por um cliente for repassada por agente *Relay* para os servidores;

Endereço *Link*-local do cliente: endereço *link-local* da interface do cliente por onde foi enviada a mensagem;
Endereço do Agente *Relay*: endereço da interface do agente *Relay* através do qual ele recebeu a mensagem *Solicit*.

Após o recebimento da mensagem *Solicit* e da verificação que está correta, o(s) servidor(es) responderá(ão) com uma mensagem *Advertise* diretamente ao cliente solicitador se pertencer(em) ao mesmo *link*, ou ao agente *Relay* que se encarregará de repassar ao cliente. O servidor utilizará como endereço IP destino o endereço *unicast* do cliente ou do agente *Relay* presentes na mensagem *Solicit* recebida. A mensagem *Advertise* tem o objetivo de informar ao cliente o endereço do agente DHCPv6 para onde a mensagem *Request* (requisição de configuração) deverá ser enviada. Além disso, também poderá enviar ao cliente informações sobre possíveis serviços e recursos que o servidor poderá disponibilizar.

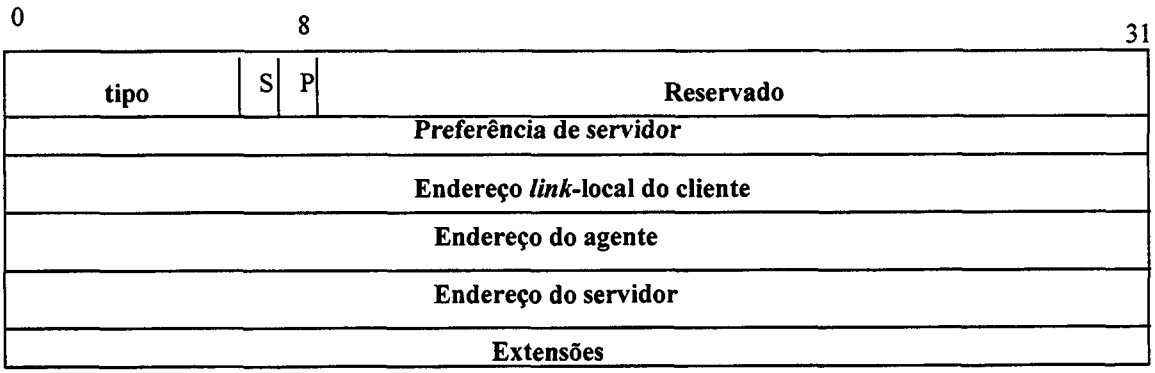


Figura 7. 14 Formato da Mensagem DHCPv6 *Advertise*

A Figura 7.14 apresenta o formato da mensagem DHCPv6 *Advertise* onde:

Tipo: possui valor 2;
S: se habilitado, o endereço do servidor está presente. Normalmente se a mensagem DHCP *Solicit* foi recebida diretamente do cliente sem intermediação de um agente *Relay* este campo estará habilitado. Caso contrário, apresentará valor zero;

P: se habilitado, a preferência de servidor é válida;

Reservado: possui valor 0;

Preferência de servidor: indica ao cliente se o servidor está pronto para fornecer os serviços ao cliente;

Endereço *Link-local* do cliente: endereço *link-local* da interface do cliente por onde foi enviada a mensagem *Solicit*;

Endereço do Agente: endereço IP de uma interface de um agente DHCP pertencente ao mesmo *link* que o cliente;

Endereço do Servidor: se presente, indica o endereço IP do servidor DHCP;

Extensões: este campo poderá estar preenchido com extensões (ver Seção 7.3.2), com o objetivo de fornecer ao cliente solicitador, as informações sobre possíveis serviços e recursos que o servidor poderá disponibilizar.

Um *host* poderá receber mensagens DHCPv6 *Advertise* de uma série de agentes, cabe a ele selecionar o servidor ou agente *Relay* para onde ele deverá enviar a mensagem de requisição de configuração DHCPv6 *Request*. Esta mensagem será enviada utilizando como endereço destino o endereço *unicast* do servidor DHCPv6. Na ausência de um servidor conectado ao mesmo *link*, o cliente enviará a mensagem ao agente *Relay* que repassará esta ao servidor.

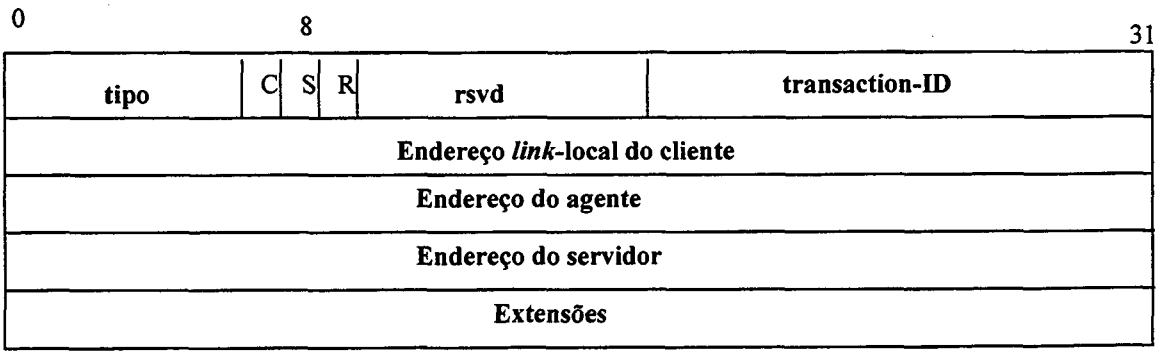


Figura 7. 15 Formato da Mensagem DHCPv6 *Request*

A Figura 7.15 mostra o formato da mensagem DHCPv6 *Solicit* onde:

Tipo: possui valor 3:

C: se habilitado, o cliente informa ao servidor para remover todos os recursos associados ao cliente, com exceção daqueles recursos citados no campo extensões. Normalmente este campo é habilitado quando o processo cliente DHCPv6 é reinicializado;

S: se habilitado, indica a presença de um servidor;

R: se habilitado, indica que o nó cliente foi reinicializado e existe a requisição para que *Transactions*-IDs associadas ao cliente sejam liberadas para reutilização;

Reservado: possui valor 0;

Transaction-ID: valor inteiro utilizado para identificar este pedido (*Request*);

Endereço *Link-local* do cliente: endereço *link-local* da interface do cliente por onde foi enviada a mensagem *Solicit*;

Endereço do Agente: endereço IP de uma interface de um agente DHCPv6 pertencente ao mesmo *link* que o cliente;

Endereço do Servidor: se presente, indica o endereço IP do servidor DHCPv6;

Extensões: este campo poderá estar preenchido com extensões (ver Seção 7.3.2), que denotam pedidos específicos de configuração feitos pelo cliente.

O servidor, por sua vez, recebe a mensagem *Request*, verifica se seu conteúdo está correto e avalia a possibilidade de disponibilização de recursos. Se, por algum motivo, os recursos requisitados pelo cliente não puderem ser contemplados, uma mensagem *Reply* será enviada reportando o problema. Por outro lado, se os recursos que devem ser atribuídos ao clientes estiverem disponíveis, uma mensagem *Reply* será enviada, contendo os parâmetros de configuração do cliente DHCPv6.

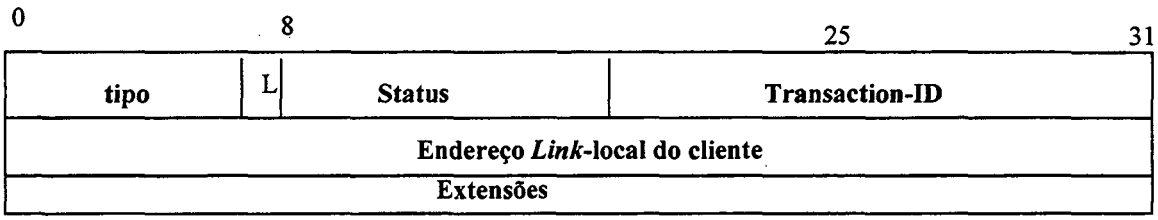


Figura 7. 16 Formato da Mensagem DHCPv6 *Reply*

O formato de mensagem DHCPv6 *Reply* é apresentado na Figura 7.16 onde:

Tipo: possui valor 4;
L: se habilitado, o endereço *link-local* do cliente está presente;
Estado: define se o pedido de requisição obteve sucesso ou não. Por exemplo, o valor 0 indica sucesso, por sua vez o valor 17 indica que não há recursos disponíveis no momento para atender os pedidos do cliente;
Transaction-ID: valor inteiro utilizado para identificar esta mensagem *Reply*. Este valor foi copiado do campo *Transaction*-ID da mensagem *Request* enviada pelo cliente;
Endereço *Link-local* do cliente: endereço *link-local* da interface do cliente por onde foi enviada a mensagem *Request*;
Extensões: extensões com os valores de configuração disponibilizados pelo servidor ao cliente (ver Seção 7.3.2).

Além das mensagens citadas anteriormente, existem mais dois tipos de mensagens utilizadas: mensagem *Reconfigure* e mensagem *Release*.

A mensagem *Reconfigure* é enviada para um cliente específico ou para todos os clientes do *link* para avisar que eles devem obter novas configurações. Os clientes então enviarão mensagens DHCPv6 *Request* para obterem seus novos padrões de configuração. O servidor irá então repassar as informações através de uma mensagem *Reply*.

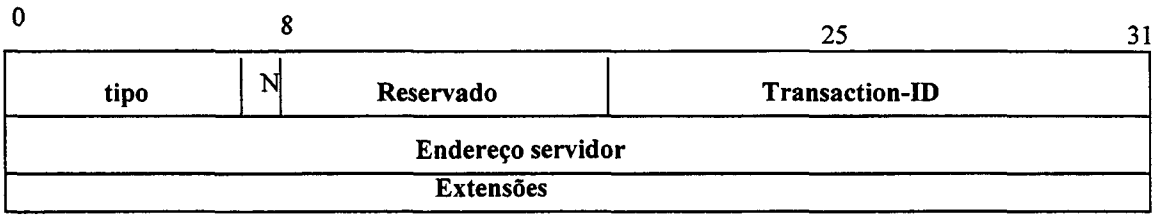


Figura 7. 17 Formato da Mensagem DHCPv6 *Reconfigure*

O formato da mensagem DHCPv6 *Reconfigure* é ilustrado na Figura 7.17 onde:

Tipo: possui valor 5;

D: se habilitado, indica que o servidor deverá enviar a mensagem *Reply* diretamente para o cliente, ao invés de utilizar um possível agente *Relay*;

Reservado: possui valor 0;

Transaction-ID: valor inteiro utilizado para identificar a mensagem *Release*;

Endereço *link-local* do cliente: endereço *link-local* da interface do cliente por onde foi enviada a mensagem *Release*;

Endereço do agente: endereço IP de uma interface de um agente DHCPv6 pertencente ao mesmo *link* que o cliente, no qual ele enviou previamente uma mensagem *Request*;

Endereço do cliente: endereço IP da interface do cliente por onde foi enviada a mensagem *Release*. Este campo estará presente sempre que o campo D tiver sido habilitado, mesmo que seu valor seja igual ao endereço *link-local*;

Extensões: extensões com os valores de configuração que o cliente gostaria de manter (ver Seção 7.3.2).

Mensagens DHCPv6 do tipo *Release*, serão emitidas por clientes aos servidores informando que todos, ou alguns de seus recursos, podem ser liberados e não devem ser mais relacionados a este cliente.

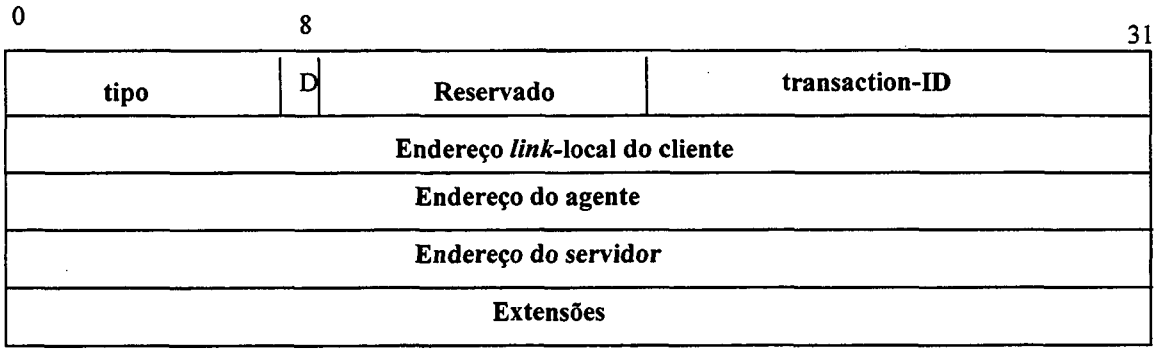


Figura 7. 18 Formato de Mensagem DHCPv6 *Release*

A Figura 7.18 apresenta o formato da mensagem DHCPv6 *Release* onde:

Tipo: possui valor 6;
N: se estiver habilitado, indica que o cliente deve esperar uma resposta DHCPv6 *Reply* em virtude de uma mensagem DHCPv6 *Reconfigure* e não de uma mensagem *Request*;
Reservado: possui valor 0;
Transaction-ID: valor inteiro utilizado para identificar esta mensagem *Reconfigure*;
Endereço do servidor: endereço IP do servidor DHCPv6 que está enviando a mensagem reconfiguração;
Extensões: extensões com os valores de configuração (ver Seção 7.3.2).

7.3.2 Extensões

A maioria das mensagens DHCPv6 apresenta o campo extensões. Este campo, contém a lista de parâmetros que o servidor DHCPv6 deverá fornecer ao cliente.

Várias são as extensões definidas, entretanto a mais importante é a extensão de endereçamento. Ela contém quatro campos importantes através dos quais serão

repassados o endereço do cliente, o tempo de vida *preferred* e *deprecate* deste endereço e nome DNS equivalente.

Além desta extensão básica, outra extensão fundamental, principalmente pelo aspecto segurança, é a *Client-Server Authentication*. Este tipo de extensão poderá estar presente durante a troca de mensagens entre um cliente e um servidor. Ela garante a autenticidade dos dados da mensagem DHCPv6.

7.3.3 Aspectos de Segurança

O cabeçalho de autenticação do IPv6 poderá fornecer segurança na troca de mensagem DHCPv6 quando cliente e servidor possuírem endereços para uma comunicação direta. Entretanto, no processo de inicialização, o cliente freqüentemente terá apenas um endereço *link-local* que não será suficiente para troca de mensagem seguras, se o servidor não se encontrar no mesmo *link* que o cliente. Neste caso será acionado o agente DHCP *Relay*, para que a mensagem obrigatoriamente seja destinada ao endereço do *Relay* presente no campo onde será preenchido com o endereço destino. No caso em que o cliente solicitante enviar a mensagem diretamente ao servidor, a extensão de autenticação deverá ser utilizada.

7.4 Atualização dos Servidores de Nomes (*DNS Update*)

O sistema de nomes originariamente foi projetado para trabalhar com uma base de dados configurada estaticamente. A alteração deste tipo de base de dados necessita a intervenção manual de um administrador, podendo na maioria das vezes representar uma considerável demora. Atualmente, a configuração de nomes é extremamente importante, principalmente na nova notação de endereçamento IP, onde fica muito trabalhoso para o usuário utilizá-la. A maioria das aplicações verifica a autenticidade de um determinado nome através das bases de dados e, se elas não apresentarem uma

coerência uma possível conexão não será efetuada. Para evitar tais transtornos, um *host* que teve seu nome alterado deverá ter este registro automaticamente alterado na base de dados DNS. Mecanismos de atualização automática já estão disponíveis (RFC 2136) e protocolos IPv6 já estão sendo implementados e projetados utilizando este conceito.

Especificamente, o método de autoconfiguração utiliza a atualização dinâmica de nomes através de sua extensão de endereçamento. É ativado por servidores DHCP ao receber mensagens DHCP *Request* e *Release* enviadas pelo cliente, onde ele envia seu nome e requisita uma atualização no DNS, ou então o recurso deve ser liberado e uma atualização no DNS se faz necessária.

7.5 *Service Location Protocol* (SLP)

O DHCPv6 tem como objetivo oferecer os parâmetros de configuração básicos necessários a um nó para que ele tenha uma completa funcionalidade na rede. Não cabe a ele fornecer todos os requisitos necessários para que um cliente possa se conectar com os diversos serviços de rede. Para esta função temos o protocolo SLP, que fornece condições para a descoberta e seleção de serviços de redes. Este protocolo elimina a necessidade de um usuário ter que conhecer o nome de um determinado *host* que suporta o serviço desejado.

O protocolo SLP é referenciado pela autoconfiguração *stateful*, através da extensão *Directory Agent*, onde é informado às entidades que utilizam o SLP, o endereço de Agentes de Diretório (*Directory Agents*), para que as transações sejam efetivadas. Os *Directory Agents* agem como centralizadores de informações para os clientes SLP, informando onde se encontram os servidores de serviço.

8 MIBS PARA AUTOCONFIGURAÇÃO DO IPV6

O protocolo IPv6 é relativamente novo se compararmos com a idade da versão anterior. Ele introduziu novas características e novas necessidades. A autoconfiguração é uma destas características. Apesar do IPv4 também possuir protocolos que permitem certo grau de configuração, nesta nova versão ela se faz essencial, sendo obrigatória a sua implementação. Por este motivo tem que apresentar excelente funcionalidade. A necessidade da autoconfiguração surgiu devido ao crescimento da Internet. Existem expectativas que alcance cada metro quadrado deste planeta [HUI97]. Seria praticamente inviável a configuração manual de cada um dos possíveis nós que a compõe.

Com os métodos de autoconfiguração, cada nó na rede terá condições de atribuir a(s) sua(s) interface(s) de rede endereço(s) IP. Entretanto, existe a necessidade da definição de outros parâmetros de configuração além do endereçamento que podem ser oferecidos por roteadores e/ou servidores, e é na configuração destas entidades é que se faz necessária a presença de um administrador de rede. Através da configuração destes roteadores e/ou servidores, é que haverá a definição de uma política básica de configuração e de controle no acesso à rede.

8.1 Motivos

Durante o trabalho de pesquisa foi constatado que o gerenciamento para protocolo IPv6 estava apenas nascendo. Antigas MIBs precisando ser reescritas para se ajustar ao novo protocolo ou então serem introduzidas. Como é clara a importância da autoconfiguração no IPv6, sendo portanto de aplicação imediata, foi decidido fazer a contribuição nesta área através da construção de MIBs de autoconfiguração.

O tamanho e a complexidade das redes faz com que o administrador tenha que ter um grau considerável de agilidade e esforço no seu gerenciamento. A intenção na construção destas MIBs é permitir que através de uma única ferramenta de

gerenciamento, roteadores e servidores de configuração possam ser configurados, avaliados e reportados, fornecendo assim a agilidade necessária ao administrador de redes.

8.2 Ambiente e Premissas

O aspecto técnico que envolveu a composição das MIBs de Autoconfiguração foi baseado no estudo do protocolo SNMP, além do estudo de várias MIBs (MIB-II, IPv6, IPv6-TCP, IPv6-UDP, IPv6-ICMP, entre outras). A composição lógica foi baseada a partir de estudos feitos dos protocolos IPv6, autoconfiguração *stateless* e DHCPv6, além de outros protocolos como SLP (*Service Location Protocol*) envolvidos em extensões DHCPv6.

O ambiente prático de estudos e validações foi composto por 3 nós, sendo 1 roteador e 2 *hosts*. Este ambiente proporcionou a compreensão prática do método de autoconfiguração *stateless*, bem como permitiu a formação de atividades simuladoras de um servidor DHCPv6 gerenciável. Fica claro, que não se contou com nenhum servidor DHCPv6 para a sua análise prática, uma vez que eles estão sendo lançados. O modelo genérico da MIB DHCPv6 foi construído baseado no seu protocolo.

As MIBs e seus respectivos agentes, bem como os módulo da aplicação-gerente foram compostos e testados utilizando as ferramentas de gerenciamento citadas no capítulo 7.

8.3 Método de Composição

O método apresentado abaixo define em linhas gerais a metodologia empregada na composição das MIBs.

O método consiste de 6 fases. Na primeira, são identificadas as variáveis fundamentais para o protocolo. Na segunda fase é feita a definição do tipo de cada uma destas variáveis. Na terceira, define-se se a variável se ajusta a um padrão escalar

ou então se ajusta a um padrão colunar, ou seja, em tabelas. Na quarta fase são definidas as variáveis de notificação.

Na quinta fase, é verificado quais destas variáveis devem ter obrigatoriedade na sua implementação. Na sexta e última fase o agente construído automaticamente com a MIB é inicializado através de ferramentas de gerenciamento e faz-se a validação. A seqüência de fases é cíclica, ou seja pode-se retornar a qualquer uma delas para completá-las ou refiná-las.

8.4 MIB *IPv6-STATELESS*

A MIB *IPv6-STATELESS* foi composta para permitir a monitoração e o controle de parâmetros de configuração que deverão ser repassados por roteadores aos *hosts* através da mensagem *Neighbor Discovery Router Advertisements*. Esta MIB foi construída baseada no método de configuração *stateless* e em experiências práticas realizadas no ambiente de testes.

8.4.1 Modelagem

A MIB *IPV6-STATELESS* conta com 2 módulos funcionais representados pela Figura 8.1.

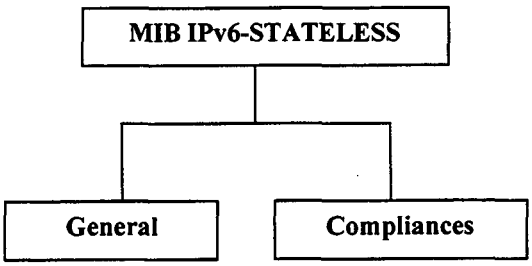


Figura 8. 1 Módulos da MIB *IPv6-STATELESS*

Módulo *General*

Pelas diferentes interfaces de um roteador, conectadas a *links* distintos, serão transmitidos os parâmetros de configuração *stateless* aos *hosts*. Os *links* necessitarão parâmetros de configuração diferentes, principalmente o prefixo de rede. Dessa forma, existe a necessidade de monitoração e controle específico para cada interface. A estratégia utilizada foi a de utilizar uma tabela, cujo número de entradas representa o número de interface, e a elas estarão associados as suas variáveis de gerenciamento.

A seguir são apresentadas as variáveis que compõe o módulo *General*:

Variável Escalar ipv6SlessInterfaces: nesta variável estará registrado o número de interfaces referentes à configuração corrente do roteador.

Variável Colunar ipv6IfSlessTable: esta variável representa uma tabela composta por "n " entradas (quantidade de entradas verificável através da variável escalar ipv6SlessInterfaces), possuindo os campos ipv6IfSlessIndex, ipv6IfSlessNdAdRcTm, ipv6IfSlessNdRtAdI, ipv6IfSlessNdRtAdI , ipv6IfSlessNdNsI, ipv6IfSlessNdSupRtAd, ipv6IfSlessNdPrefixAd , ipv6IfSlessNdValLt, ipv6IfSlessManAddConfFlag, ipv6IfSlessOtherStatefulConfFlag, ipv6IfSlessMtu que por sua vez, são representados através de variáveis escalares com os mesmos nomes descritos abaixo.

Variável Escalar ipv6IfSlessIndex: esta variável conterá valor de 1 até ipv6SlessInterfaces identificando a interface nesta tabela;

Variável Escalar ipv6IfSlessNdAdRcTm: configura o tempo que um nó remoto é tido como alcançável depois que alguma confirmação de alcançabilidade tenha sido identificada. O valor desta variável deve ser

incluída em todas as mensagens *Router Advertisement* enviadas através desta interface.

Variável Colunar ipv6IfSlessNdRtAdI : configura o intervalo de tempo entre as transmissões *Router Advertisements* que são enviadas através desta interface.

Variável Escalar ipv6IfSlessHl: configura o limite de *hop* do cabeçalho IP dos pacotes a serem enviados pelos *hosts*. O valor desta variável deve ser incluída em todas as mensagens *Router Advertisement* enviadas através desta interface.

Variável Escalar ipv6IfSlessNdRtAdLt: configura o tempo de vida de uma mensagem *Router Advertisement*. O valor desta variável deve ser incluída em todas as mensagens *Router Advertisement* enviadas através desta interface.

Variável Escalar ipv6IfSlessNdNsI: configura o intervalo de tempo entre mensagens *Solicit*. O valor desta variável deve ser incluída em todas as mensagens *Router Advertisement* enviadas através desta interface.

Variável Escalar ipv6IfSlessNdSupRtAd: controla a transmissão de *Router Advertisements* na interface. O valor 1 (*off*) indica que através da interface não serão enviadas mensagens *Router Advertisements*. Já o valor 2 (on) indica o contrário.

Variável Escalar ipv6IfSlessNdPrefixAd: determina explicitamente quais prefixos de rede devem ser transmitidos em mensagens *Router Advertisements*. Se esta variável não tiver valor atribuído, então o roteador irá transmitir os prefixos utilizados pela interface.

Variável Escalar ipv6IfSlessNdPrefixLength: define o tamanho do prefixo de rede.

Variável Escalar ipv6IfSlessManAddConfFlag: O valor 1 (*off*) indica que as mensagens *Router Advertisements* a serem enviadas deverão estar com o *flag Managed Address Configuration* habilitado, indicando aos *hosts* que deverão utilizar configuração *stateful* para obtenção de endereço IP. Já o valor 2 (on) indica que as mensagens *Router Advertisements* a serem enviadas deverão estar com o *flag Managed Address Configuration* desabilitado, permitindo aos *hosts* que utilizem autoconfiguração *stateless* para a criação de endereços *unicast* globais.

Variável Escalar ipv6IfSlessOtherStatefulConfFlag; O valor 1 (off) indica que as mensagens *Router Advertisements* a serem enviadas deverão estar com o *flag Other Stateful Configuration* habilitado, indicando aos *hosts* que deverão utilizar configuração *stateful* para a obtenção de outros parâmetros de configuração com exceção do endereço.

Variável Escalar ipv6IfSlessMtu: configura o valor da MTU definida para o *link*. O valor desta variável deve ser incluída em todas as mensagens *Router Advertisements*

Módulo *Compliances*

Este módulo define o limite mínimo de acesso que deve ser empregado na implementação desta MIB. Normalmente este procedimento é adotado para determinar quais variáveis com acesso de leitura e escrita como limite máximo possa ser implementadas apenas com limite mínimo ou seja, apenas de leitura[1]. A Figura 8.2 ilustra

[1] O limite de acesso de uma variável é determinado pelo elemento de acesso MAX-ACCESS (ver ANEXO 1)

a definição de uma variável que possui como valor de acesso *read-write* e que quando implementada terá que ter o valor mínimo de *read-only*.

```
                Definição da Variável na MIB
ipv6IfSlessMtu     OBJECT-TYPE
      SYNTAX       Integer32
      MAX-ACCESS   read-write
      STATUS       current
::=   {  ipv6IfSlessEntry  12  }


 Limite de Acesso Mínimo definido para implementações

OBJECT ipv6IfSlessMtu
                   MIN-ACCESS read-only
```

Figura 8.2 Definição do Limite de Acesso Mínimo de uma Variável

No caso específico da MIB *IPV6-STATELESS* desenvolvida, as variáveis apresentadas na Figura 8.3 têm por definição o limite de acesso máximo estabelecido como *read-write*. Entretanto, fica definido através do módulo *Compliances* que o limite mínimo que poderá ser utilizado será *read-only*. As demais variáveis não citadas, já possuem o limite máximo definido como *read-only*, não necessitando que sejam abordadas.

ipv6IfSlessIndex, ipv6IfSlessNdAdRcTm,
ipv6IfSlessNdRtAdI, ipv6IfSlessHl,
ipv6IfSlessNdRtAdLt, ipv6IfSlessNdNsI,
ipv6IfSlessNdSupRtAd,
ipv6IfSlessNdPrefixAd,
ipv6IfSlessNdPrefixLength,
ipv6IfSlessManAddConfFlag,
ipv6IfSlessOtherStatefulConfFlag,
ipv6IfSlessMtu

Figura 8.3 Variáveis da MIB *IPV6-STATELESS* com Acesso *read-write*

8.4.2 Experimentos e Simulações

Os experimentos para validar a MIB *IPV6-STATELESS* foram realizados utilizando a ferramenta *NetMonitor*, através de sua aplicação *MIB Browser* e da ferramenta *AgentBuilder*, através da execução do agente. O agente residiu em uma máquina IBM RS6000 com sistema operacional AIX, utilizando a porta 161 para o recebimento das mensagens SNMP. A porção gerente (MIB *Browser*), residiu em um IBM PC com sistema operacional Windows NT. Outro ponto importante é que estas ferramentas ainda não disponibilizam conexões IPv6, sendo portanto, utilizado endereçamento IPv4 durante a troca de mensagens. A ferramenta *NetMonitor* MIB *Browser* permite que as MIBs envolvidas diretamente com a MIB *IPV6-STATELESS*, além dela própria sejam carregadas. Dessa forma as informações disponibilizadas através de um agente podem ser monitoradas e controladas. A Figura 8.4 mostra a carga das MIBs SNMPv2-TC, SNMPv2-MIB, IPV6-TC e *IPV6–STATELESS*, permitindo assim que as variáveis definidas na MIB *IPV6-STATELESS* seja acessadas.

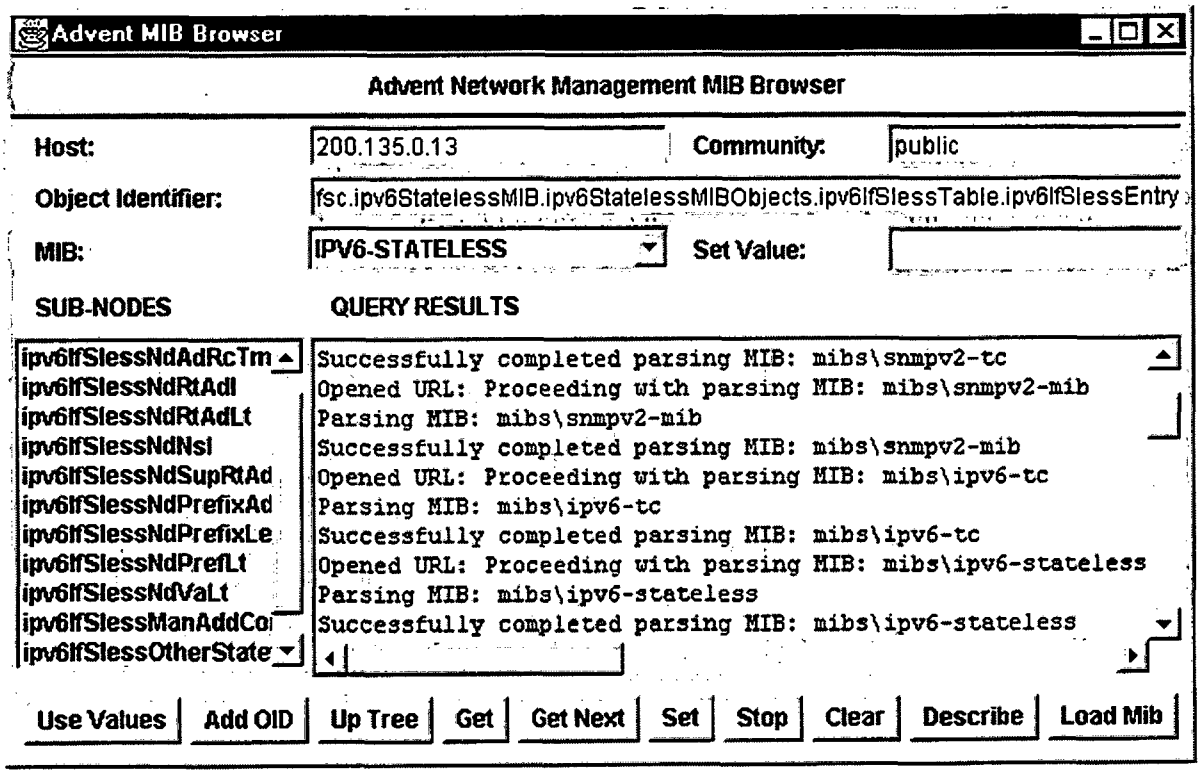


Figura 8. 4 MIB *IPV6-STATELESS* sendo carregada através da *AdventNet MIB Browser*

Depois da carga das MIBs, várias operações foram realizadas envolvendo todas as variáveis da MIB *IPV6-STATELESS*. A cada operação, além da verificação através da ferramenta de monitoração, um software de simulação era executado para checar a validade das informações recebidas através do *NetMonitor* MIB *Browser*. A Figura 8.5 apresenta as duas operações envolvidas nesta MIB, ou seja de *get* (*GetRequest*) e *set* (*SetRequest*) respectivamente em uma tabela SNMP. Nesta figura são mostradas três operações. Na primeira, é realizado um comando *get* para recuperar o conteúdo da variável representada pelo campo *Object Identifier*. A segunda operação é um realizado um *set* com o valor indicado no campo *Set Value* em seguida é realizada uma operação *get* para checar se houve a alteração requisitada.

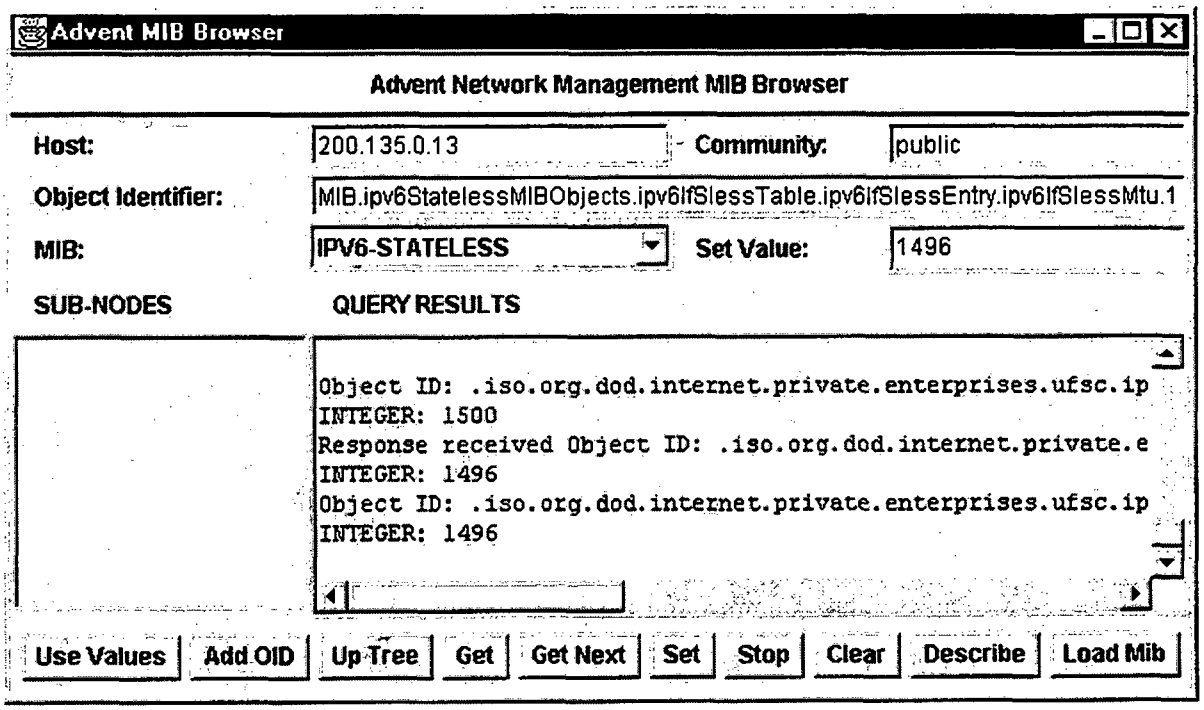


Figura 8. 5 Exemplo de Operações *GetRequest* e *SetRequest*

Como dito anteriormente, depois que operações de *set* eram executadas, fazia-se a checagem através de uma aplicação, chamada **showip6**, que apresenta as variáveis envolvidas no processo de autoconfiguração *stateless*. A Figura 8.6 mostra a saída da execução deste programa.

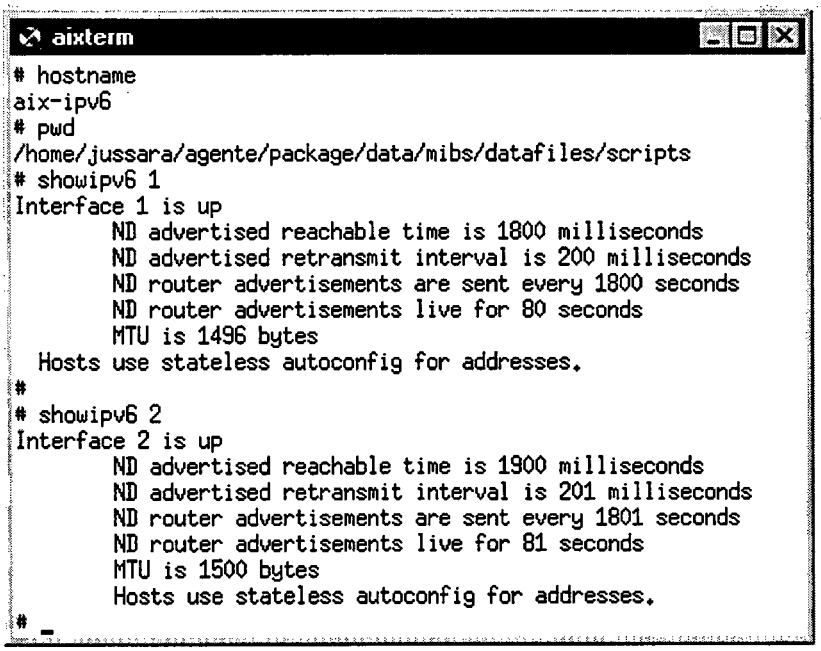


Figura 8. 6 Saída da Aplicação showipv6

8.5 MIB DHCPv6

A MIB DHCPv6 foi composta para permitir a monitoração e o controle de parâmetros de configuração que deverão ser repassados por servidores DHCPv6 aos seus clientes através da troca de mensagem DHCPv6. Esta MIB foi construída baseada no método de configuração *stateless* e do protocolo.

8.5.1 Modelagem

A MIB DHCPv6 conta com 8 módulos distintos representados pela Figura 8.7.

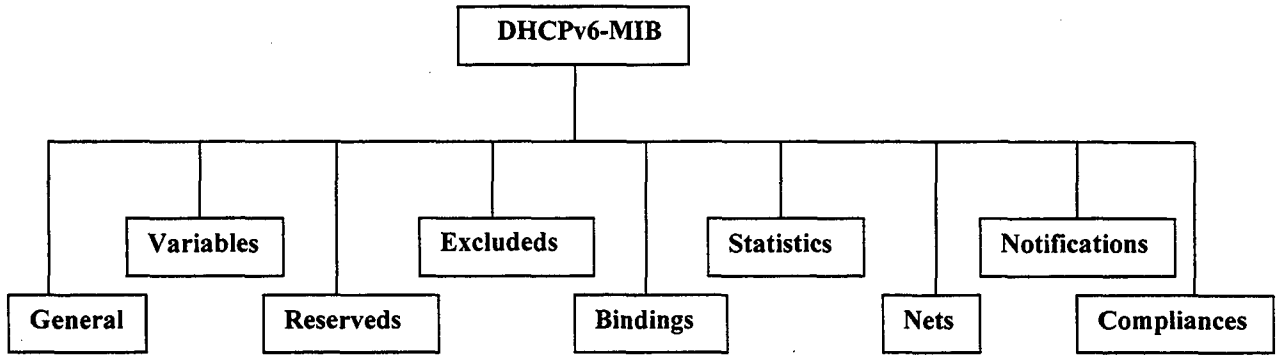


Figura 8. 7 Módulos da MIB DHCPv6

A seguir serão apresentados os módulos da MIB DHCPv6:

Módulo *General*

Este módulo contém duas variáveis, upTime e operation, que permitem verificação, ação inicializadora e finalizadora do servidor DHCPv6.

Variável Escalar upTime: esta variável representa o tempo desde a última inicialização do servidor. Através desta variável o administrador terá condições de saber se a o processo DHCPv6 vem se mantendo ativo por tempos desejáveis. Se não existe, a imprudência na sua utilização, desligamentos voluntário e e/ou involuntários acontecendo freqüentemente podendo por em risco a estabilidade de um ou mais links. Quando o agente não estiver inicializado o valor desta variável será zero.

Variável Escalar operation: é permitido através desta variável que o administrador remotamente inicialize e finalize o servidor DHCPv6. Mensagens de notificação serão enviadas em tentativas de inicialização, quando este servidor já está inicializado. Da mesma forma para a finalização. Havendo uma tentativa de finalização com o servidor não inicializado haverá o envio de uma mensagem reportando a causa. O

administrador que tiver pleno conhecimento da MIB DHCPv6 verificará de antemão a variável descrita anteriormente;

Módulo *Variables*

O objetivo deste módulo é permitir a configuração e/ou a verificação do conteúdo das variáveis de tempo que são utilizadas no processo de transmissão e retransmissão de mensagens DHCPv6 emitidas pelo servidor

Variável Escalar varSMinAd: esta variável SNMP representa a variável DHCPv6 SERVER_MIN_ADV_DELAY, que por sua vez determina a quantidade mínima de tempo que um servidor aguarda para transmitir uma mensagem DHCPv6 *Advertise* depois de receber uma mensagem DHCPv6 *Solicit* enviada por um cliente.

Variável Escalar varSMaxAd: esta variável SNMP representa a variável DHCPv6 SERVER_MAX_ADV_DELAY, que por sua vez determina a quantidade máxima de tempo que um servidor aguarda para transmitir uma mensagem DHCPv6 *Advertise* depois de receber uma mensagem DHCPv6 *Solicit* enviada por um cliente.

Variável Escalar varRecMsgTimeout: esta variável SNMP representa a variável DHCPv6 RECONF_MSG_TIMEOUT, que por sua vez determina tempo em segundos que um servidor aguarda para receber uma mensagem DHCPv6 *Request* enviada por um cliente.

Variável Escalar varRecMsgMinRet: esta variável SNMP representa a variável DHCPv6 RECONF_MSG_MIN_RETRANS, que por sua vez determina o número mínimo de mensagens DHCPv6 *Reconfigure* que um servidor deve retransmitir.

Variável Escalar varRecMsgRetI: esta variável SNMP representa a variável DHCPv6 RECONF_MSG_ RETRANS_INTERVAL, que por sua vez determina o tempo mínimo que deve existir entre sucessivas retransmissões de mensagens DHCPv6 *Reconfigure* enviadas por um servidor.

Variável Escalar varXidTimeout: esta variável SNMP representa a variável DHCPv6 XID_TIMEOUT, que por sua vez determina a quantidade de tempo que um servidor DHCPv6 tem para encontrar a identificação de uma transação (*transaction* ID) de um cliente, para determinar que retransmissões feitas pelo cliente que utilizem a mesma *transaction* ID, são idempotentes.

Módulo *Reserveds*

Este módulo é composto por três sub-módulos e tem como objetivo dar condições ao administrador de rede de adicionar, excluir e verificar endereço ou grupos de endereços que são considerados reservados, devido às necessidades administrativas. Ou seja, certos endereços podem estar selecionados com finalidades específicas, devendo ser atribuído exclusivamente a uma máquina ou um grupo delas. É importante salientar que o DHCPv6 permite que endereços sejam atribuídoa a máquinas específicas.

Módulo *addReserved*

Será através deste módulo que endereços serão reservados a hosts específicos.

Variável Escalar addReservedLinkLocalAddress: através desta variável o administrador especificará o endereço *link-local* do host a que se deseja atribuir o endereço reservado.

Variável Escalar addReservedPrefixLength: determina o tamanho do prefixo de rede do endereço que deverá participar da lista de endereços reservados.

Variável Escalar addReservedAddress: endereço de rede que será atribuído ao host com o endereço *link-local* especificado através da variável escalar addReservedLinkLocalAddress .

Variável Escalar addReservedAct: através desta variável é que efetivamente acontecerá a atribuição de um endereço reservado a um host específico. Está variável conterá duas opções *add* e *clear*. Se a opção for por *clear* os valores serão removidos das variáveis e o procedimento será cancelado; entretanto se a opção for feita por *add* será feito de antemão a verificação da consistência dos valores atribuídos às variáveis addReservedPrefixLength, e addReservedAddress; havendo consistência o procedimento procegue com o registro; não havendo consistência uma notificação será enviada, sugerindo que o procedimento seja refeito. Se por ventura, este endereço reservado tenha sido atribuído dinamicamente a um host, uma mensagem DHCPv6 *Reconfigure* deverá ser enviada para que ele receba um novo endereço.

Módulo *reservedTable*

Este módulo permitirá que seja verificado através de uma tabela a relação de endereços reservados e seus respectivos hosts.

Variável Escalar reserveds: nesta variável estará registrado o número de endereços reservados, segundo a configuração corrente do servidor DHCPv6;

Variável Colunar reservedTable: esta variável representa um tabela composta por "n " entradas (quantidade de entradas verificável através da variável escalar reserveds descrita anteriormente), possuindo os seguintes

campos reservedIndex, reservedLinkLocal, reservedPrefixLength, reservedAddress, que por sua vez são representados através de variáveis escalares com o mesmo nome descritos abaixo;

Variável Escalar reservedIndex: está variável conterá um valor de 1 até reserveds, identificando o endereço reservado nesta tabela;

Variável Escalar reservedLinkLocal: esta variável contém o endereço *link-local*, da máquina a que foi reservado o endereço;

Variável Escalar reservedPrefixLength: contém o valor do prefixo de rede do endereço reservado;

Variável Escalar reservedAddress: contém o valor do endereço de rede que foi atribuído ao host com o endereço *link-local,* verificado na variável escalar reservedLinkLocal.

Módulo *deleteReserved*

Através deste módulo é que um endereço deixará de ser reservado a um determinado host. Este módulo apresenta apenas uma variável, descrita a seguir:

Variável Escalar deleteReserved: o administrador atribuirá uma valor a esta variável baseado na entrada da tabela de endereços reservados. Por exemplo, se entrar com o valor 2, será removida o endereço reservado correspondente à segunda entrada na tabela de endereços reservados. Após esta remoção deverá ser enviado um mensagem DHCPv6 *Reconfigure* ao cliente, para que solicite uma nova configuração.

Módulo *Excludeds*

Este módulo é composto por três sub-módulos e tem como objetivo dar condições ao administrador de rede de adicionar, excluir e verificar endereços ou grupos de endereços que são considerados excluídos, isto é, endereços gerenciáveis pelo servidor DHCPv6 que não deverão ser atribuídos dinamicamente.

Módulo *addExcluded*

Através deste módulo será feito o registro de endereços que serão excluídos do grupo de endereços gerenciáveis pelo servidor DHCPv6 em questão. Poderá ser feita a exclusão de um grupo de endereços contínuos, bem como a exclusão de um único endereço.

Variável Escalar addExcludedPrefixLength: o administrador deverá entrar com o valor do prefixo de rede do endereço a ser reservado;

Variável Escalar addExcludedRangeBegin: variável que conterá depois de sua atribuição o endereço de menor valor do grupo de endereços contínuos a serem removidos;

Variável Escalar addExcludedRangeEnd: está variável conterá o endereço de mais alto valor do grupo de endereços a ser removido. Se existir a necessidade da exclusão de um único endereço e não de uma fatia de endereçamento, então o conteúdo da variável addExcludeRangeEnd será o mesmo que o atribuído à variável addExcludeRangeBegin;

Variável Escalar addExcludedAct: através desta variável é que efetivamente acontecerá a exclusão de um endereço ou de um grupo de endereços. Está variável conterá duas opções *add* e *clear*. se a opção for por *clear* os valores serão removidos das variáveis e o procedimento será cancelado; entretanto

se a opção for feita por *add,* será feito de antemão a verificação da consistência e validade dos valores atribuídos às variáveis addExcludedRangeBegin, addExcludedRangeEnd; havendo consistência o procedimento procegue com o registro; não havendo consistência, uma notificação será enviada, sugerindo que o procedimento seja refeito.

Módulo *excludedTable*

Este módulo permitirá que seja verificado através de uma tabela a relação dos endereços excluídos gerenciáveis pelo servidor DHCPv6

Variável Escalar excludeds: nesta variável estará registrado o número de endereços ou grupos de endereços excluídos segundo a configuração corrente do servidor DHCPv6;

Variável Colunar excludedTable: esta variável representa um tabela composta por "n " entradas (quantidade de entradas verificável através da variável escalar excludeds descrita anteriormente), possuindo os seguintes campos excludedIndex, excludedPrefixLength, excludedRangeBegin, excludedRangeEnd , que por sua vez são representados através da variáveis escalares com o mesmo nome descritas abaixo;

Variável Escalar excludedIndex: está variável conterá um valor de 1 até excludeds, identificando o endereço ou grupo de endereços excluídos nesta tabela;

Variável Escalar excludedPrefixLength: identifica o tamanho do prefixo de rede do endereço ou grupo de endereços excluídos, referentes a esta entrada na tabela;

Variável Escalar excludedRangeBegin: identifica o endereço de menor valor do grupo de endereços contínuos que foram excluídos, referentes a esta entrada na tabela;

Variável Escalar excludedRangeEnd: identifica o endereço de maior valor do grupo de endereços contínuos que foram excluídos, referentes a esta entrada na tabela.

Módulo *deleteExclude*

Através deste módulo é que um endereço ou uma fatia de endereços deixará de ser excluída. Este módulo apresenta apenas uma variável, descrita a seguir:

Variável Escalar deleteExcluded: o administrador atribuirá um valor a esta variável, baseado na entrada da tabela de endereços excluídos. Por exemplo, se entrar com o valor 2, será removida o endereço ou fatia de endereços excluídos correspondente a segunda entrada na tabela de endereços excluídos.

Módulo *Bindings*

O módulo *bindings* permite a determinação de quantos clientes receberam informações de configuração do servidor DHCPv6 em questão, bem como possibilita que se verifique o conteúdo das informações repassadas. Além disto, através deste módulo será permitido ao administrador detectar a atividade ou não de um cliente.

Variável Escalar bindings: nesta variável estará registrado o número de clientes que se autoconfiguraram a partir de informações de configuração delegadas pelo servidor DHCPv6 em questão;

Variável Colunar bindingTable: esta variável representa um tabela composta por "n " entradas (quantidade de entradas verificável através da variável escalar bindings descrita anteriormente), possuindo uma série de campos, sendo representados através das variáveis escalares descritas abaixo;

Variável Escalar bindingIndex: está variável conterá um valor de 1 até bindings, identificando o cliente que recebeu uma configuração do servidor DHCPv6;

Variável Escalar bindingLynkLocalAddress: identifica o endereço *link-local* do cliente DHCPv6;

Variável Escalar bindingAgentAddressPrefix: identifica o endereço do agente (Servidor ou Relay) que está diretamente conectado ao mesmo link do cliente;

Variável Escalar bindingHostId: porção do endereço que representa o host. Este valor pode ou não ter sido atribuído pelo servidor DHCPv6, uma vez que se foi adotada a configuração *stateless other stateful configuration*, este campo terá o valor do identificador de interface.

Variável Escalar bindingLeasedTime: tempo de vida determinado que um cliente poderá utilizar esta configuração;

Variável Escalar bindingNameServers: identifica os servidores DNS (*Domain Name Service*) definidos para este cliente;

Variável Escalar bindingDomainName: identifica o domínio DNS definido para este cliente;

Variável Escalar bindingDirAgent: identifica o endereço do *Directory Agent* definidos para este cliente para que ele possa utilizar o SLP (*Service Location Protocol*);

Variável Escalar bindingDirAgentScopeList: define a lista dos tipos de serviços que podem ser oferecidos por um mesmo nó na rede. Esta associado diretamente à variável bindingDirAgent;

Variável Escalar bindingServiceScope: indica um escopo que deve ser utilizado por um *Service Agent*;

Variável Escalar bindingNtpServers: identifica o servidores NTP (*Network Time Protocolo*) definidos para este cliente;

Variável Escalar bindingNisDomainName: identifica o domínio NIS definido para este cliente;

Variável Escalar bindingsNisServes: identifica os servidores NIS definidos para este cliente;

Variável Escalar bindingNisPlusDomainName: identifica o domínio NIS+ definido para este cliente;

Variável Escalar bindingNisPlusServers: identifica o servidores NIS+ definidos para este cliente;

Variável Escalar bindingsKeepAliveInternal: especifica o intervalo em segundos que um cliente TCP deve esperar antes de enviar um mensagem *keepalive*;

Variável Escalar bindingActivitie: através desta variável o administrador de rede terá condições de verificar se o cliente representado por este *binding* está ativo ou não.

Módulo *Statistics*

Este módulo fornece valores estatísticos relacionados diretamente com a mensagens DHCPv6 trocadas entre o servidor e seus clientes.

Variável Escalar statSolicit: determina o número de mensagens *Solicit* recebidas pelo servidor DHCPv6;

Variável Escalar statSolicitError: determina o número de mensagens *Solicit* com conteúdo inadequado recebidas pelo servidor DHCPv6;

Variável Escalar statAdvertise: determina o número de mensagens *Advertise* enviadas pelo servidor DHCPv6;

Variável Escalar stat*Request*: determina o número de mensagens *Request* recebidas pelo servidor DHCPv6;

Variável Escalar statReply: determina o número de mensagens *Reply* enviadas pelo servidor DHCPv6;

Variável Escalar statReplyUnspError: determina o número de mensagens *Reply* enviadas, que indicam ao cliente que houve uma falha na transação, cujo motivo não foi identificado;

Variável Escalar statReplyAuthError: determina o número de mensagens *Reply* enviadas, que indicam ao cliente que houve uma falha de Autenticação;

Variável Escalar statReplyFormedError: determina o número de mensagens *Reply* enviadas, que indicam ao cliente que sua mensagem *Request* ou *Release* enviada estava mal formada;

Variável Escalar statReplyResourcesError: determina o número de mensagens *Reply* enviadas, que indicam ao cliente que o recurso solicitado não está disponível;

Variável Escalar statReplyClientRecError: determina o número de mensagens *Reply* enviadas pelo servidor em que os recursos requisitados pelo clientes não podem ser oferecidos;

Variável Escalar statReplyInvalidClientIpError: determina o número de mensagens *Reply* enviadas, que indicam ao cliente que o endereço especificado na mensagem *Release* é um endereço inválido;

Variável Escalar statReplyCharSetError: determina o número de mensagens *Reply* enviadas, que indicam ao cliente que o servidor não pode entender o conteúdo do grupo de caracteres selecionados;

Variável Escalar statRelease: determina o número de mensagens *Release* recebidas pelo servidor DHCPv6;

Variável Escalar statReleaseError: determina o número de mensagens *Release* com conteúdo inadequado recebidas pelo servidor DHCPv6;

Variável Escalar statReconfigure: determina o número de mensagens *Reconfigure* enviadas pelo servidor DHCPv6;

Módulo *Notifications*

As notificacões ou *traps* serão utilizadas para alertar o administrador de rede, através de uma ferramenta de gerenciamento, que situações ou atividades críticas estão sendo realizadas. Podendo estas, segundo a visão do gerente, serem de caráter totalmente informacional, ou então serem alertas de possíveis atitudes indevidas realizadas por pessoas com permissão ou não de acesso ao servidor DHCPv6.

Notificação dhcpv6StatusChange: este tipo de notificação será enviada ao gerente SNMP, toda vez que houver uma mudança no estado do servidor ou seja quando for finalizado ou inicializado;

Notificação dhcpv6AddReservedActAdd: toda vez que um endereço for reservado uma notificação será enviada ao gerente SNMP. Esta notificação deverá conter os dados do endereço reservado;

Notificação dhcpv6DeleteReservedActAdd: toda vez que um endereço deixar de ser reservado a um determinado host uma notificação será enviada ao gerente SNMP. Esta notificação deverá conter os dados do endereço que deixou de reservado;

Notificação dhcpv6AddExludeActAdd: esta notificação será enviada ao gerente SNMP quando um endereço ou uma fatia de endereços for definida para não ser repassada como valor de configuração aos hosts clientes. Como conteúdo da notificação deverá ser enviado os dados do endereço ou grupo de endereços excluídos.

Notificação dhcpv6DeleteExcludedActAdd: toda vez que um endereço ou grupo de endereços deixar de pertencer a lista de endereços excluídos uma notificação será enviada ao gerente SNMP. Como conteúdo da notificação,

deverá ser enviado os dados do endereço ou grupo de endereços que deixaram de ser excluídos.

Notificação dhcpv6BindingActivitieActivitieChange: toda vez que o servidor DHCPv6 identificar a mudança de estado de um cliente uma notificação deverá ser enviada ao gerente SNMP;

Notificação dhcpv6StatReplyAuthError: tentativas de obter configuração que foram não completadas devido a uma falha na autenticação deverão ser automaticamente reportadas, sendo que esta notificação deverá conter o endereço *link-local* do cliente bem com o endereço do agente DHCPv6;

Módulo *Nets*

Este módulo é composto por três sub-módulos e tem como objetivo dar condições ao administrador de rede de adicionar, excluir e verificar os parâmetros de configuração que devem ser atribuídos aos clientes DHCPv6.

Módulo *netTable*

O sub-módulo *netTable* tem o objetivo de verificar a configuração global de um servidor DHCPv6. Como um servidor poderá ser responsável pelo fornecimento de informações de configuração de um ou mais *links*, podendo estabelecer para um mesmo *link* mais de um padrão de configuração, cada um destes padrões, foram agrupados neste projeto em estruturas chamadas *Nets*. Cada uma destas *Nets* terá então valores de configuração distintos.

Este módulo apresenta duas variáveis, sendo uma escalar e outra colunar, esta última sendo composta por um conjunto de dezoito variáveis escalares. A utilização deste módulo introduz algumas mudanças em módulos anteriores que serão apontadas no final desta seção.

Variável Escalar nets: nesta variável estará registrado o número de *Nets* presentes na corrente configuração do servidor DHCPv6;

Variável Colunar netTable: esta variável representa um tabela composta por "n " entradas (quantidade de entradas verificável através da variável escalar nets descrita anteriormente), cada uma destas entradas possui 18 campos que representam a configuração de cada uma das *Nets*;

Variável Escalar netIndex: está variável conterá um valor de 1 até nets, identificando a *Net* que se deseja verificar a configuração;

Variável Escalar netDesc: apresenta uma descrição sucinta da *Net* em questão;

Variável Escalar netRangeBegin: variável que conterá o endereço de menor valor do grupo de endereços contínuos atribuídos a esta *Net*;

Variável Escalar netRangeEnd: variável que conterá o endereço de maior valor do grupo de endereços contínuos atribuídos a esta *Net*;

Variável Escalar netLeasedTime: tempo de vida de configuração que é delegado ao clientes desta *Net*;

Variável Escalar netLastChange: informa o tempo desde a última atualização;

Variável Escalar netNameServers: identifica o servidores DNS (*Domain Name Service*) definidos para esta *Net*;

Variável Escalar netDomainName: identifica o domínio DNS definido para esta *Net*;

Variável Escalar netDir*Agent*: identifica o endereço do *Directory Agent* definidos para esta *Net* para que ele possa utilizar o SLP (*ServiceLocation Protocol*);

Variável Escalar netDirAgentScopeList: define a lista dos tipos de serviços que podem ser oferecidos por um mesmo nó na rede. Está associada à variável netDir*Agent*;

Variável Escalar netServiceScope: indica um escopo que deve ser utilizado por um *Service Agent*;

Variável Escalar netNtpServers: identifica o servidores NTP (*Net*work *Time Protocol*) definidos para esta *Net*;

Variável Escalar netNisDomainName: identifica o domínio NIS definido para esta *Net*;

Variável Escalar netNisServers: identifica o servidores NIS definidos para esta *Net*;

Variável Escalar netNisPlusDomainName: identifica o domínio NIS+ definido para esta *Net*;

Variável Escalar netNisPlusServers: identifica o servidores NIS+ definidos para esta *Net*;

Variável Escalar netKeepAliveInternal: especifica o intervalo em segundos que um cliente TCP deve esperar antes de enviar um mensagem *keepalive*;

Variável Escalar netAdmStatus: permite a verificação do estado desta *Net*, ou seja se esta ativa ou não;

Módulo *addNet*

Através deste módulo será feita a inclusão de uma *Net*, com todos os parâmetros de configuração desejáveis. As variáveis definidas neste módulo são equivalentes aquelas apresentadas no módulo netTable, com exceção das variáveis netIndex e netAdmStatus e podem ser verificadas no Anexo2.

Módulo *deleteNet*

Através deste módulo é que será feita a exclusão de uma *Net*. Este módulo apresenta apenas uma variável, descrita a seguir:

Variável Escalar deleteNet: o administrador atribuirá uma valor a esta variável baseado na entrada da tabela de *Nets*. Por exemplo, se entrar com o valor 2, a *Net* corresponde à segunda entrada na tabela de *Nets*.

Como dito anteriormente, com a integração do módulo *Net,* será necessário a alteração dos módulos Reserveds, Excludeds em seu respectivos sub-módulos addRerved, reservedTable, addExcluded, excludedTable e a identificação de qual *Nets* estes endereços estão sendo referenciados. Abaixo as modificações propostas:

Módulo *Reserveds*

Módulo *addReserved*

Variável Escalar addReserverdNetId: este valor definirá de que *Net* este endereço foi reservado;

Módulo *reservedTable*

Variável Escalar reserverdNetId: permite verificar a que *Net* este endereço pertence.

Módulo *Excludeds*

Módulo *addExcluded*

Variável Escalar addExcludedNetId: este valor definirá de que *Net* este endereço ou fatia de endereços foram excluídos;

Módulo *excludedTable*

Variável Escalar addExcludedNetId: permite verificar a que *Net* este endereço ou grupos de endereços excluídos pertence;

Módulo *Compliances*

Este módulo foi construído com o mesmo objetivo do módulo *Compliances* da MIB *IPV6-STATELESS* (ver Seção 8.2.1 – Módulo *Compliances*).

8.5.2 Experimentos e simulações

Para ilustrar a utilização e a coerência das MIBs desenvolvidas foram elaborados alguns módulos de gerenciamento, além disso, teste similares aos realizados com a MIB *IPV6-STATELESS* também foram realizados. A Figura 8.8 mostra a carga da MIB DHCPv6 através da ferramenta *AdventNet MIB Browser*.

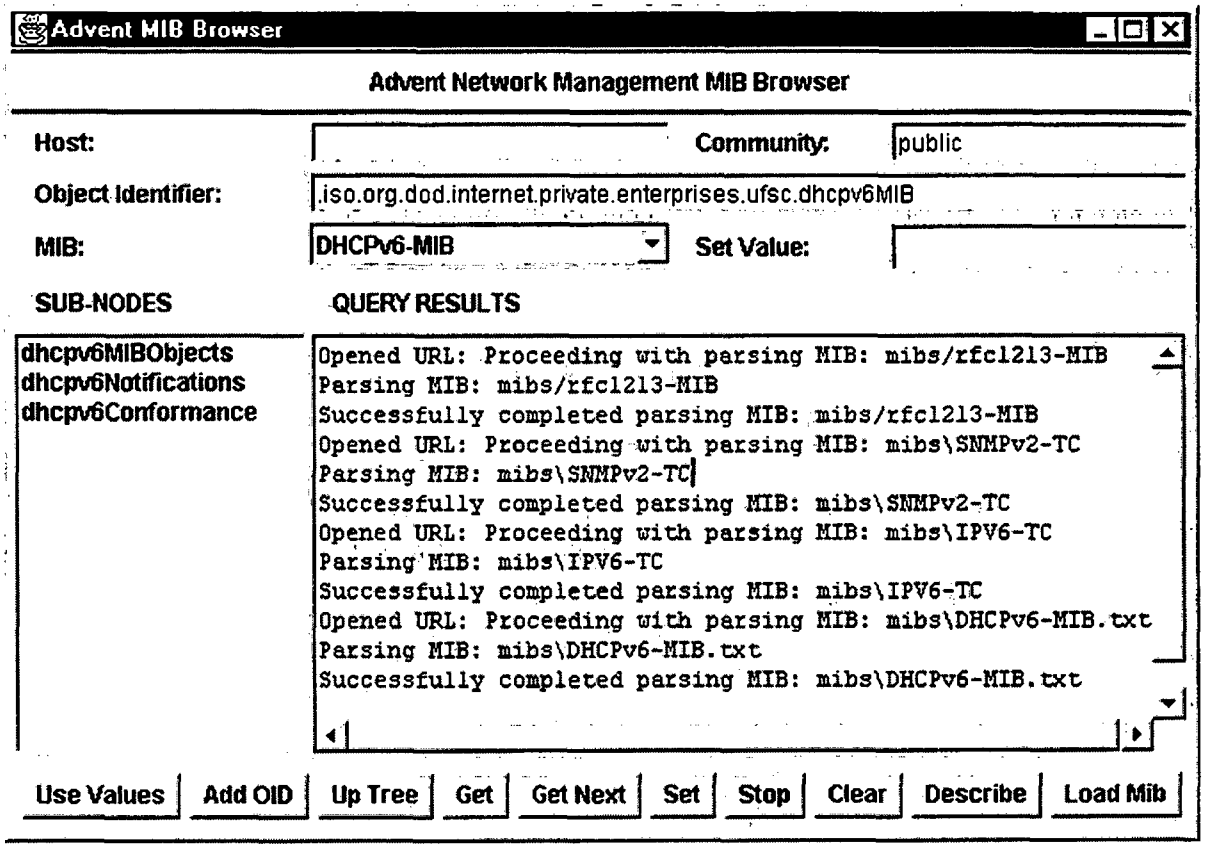


Figura 8. 8 MIB DHCPv6 sendo carregada através da *AdventNet MIB Browser*

A seguir serão apresentados alguns módulos de gerenciamento desenvolvidos durante a fase de testes da MIB DHCPv6.

Módulo 1 – Grupo *General*

Este módulo foi criado para testes do módulo *General* desenvolvido na MIB DHCPv6. Através deste módulo são executadas três operações ativadas pelo gerente e mais uma mensagem gerada pelo agente. As operações realizadas são aquelas permitidas pelas variáveis do Módulo *General* definidas na MIB DHCPv6. A Figura 8.9 mostra o módulo de gerenciamento desenvolvido para gerenciar as variáveis do Módulo *General.*

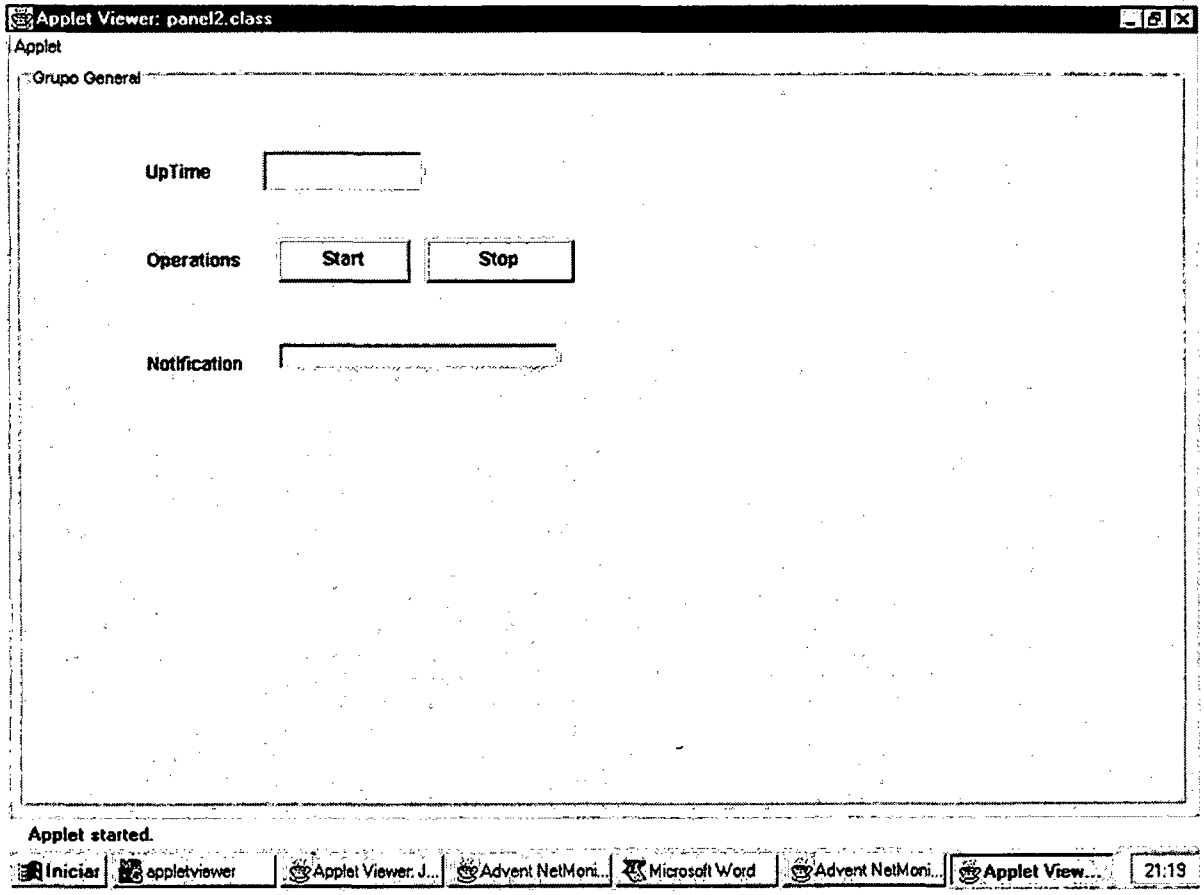


Figura 8. 9 Gerenciamento utilizando Variáveis do Módulo *General*

Módulo 2 – Grupo *Reserveds*

A construção deste módulo teve como objetivo validar as ações de configuração propostas pela MIB DHCPv6 em seus módulo *Reserveds* e *Excludeds*, uma vez que são bastante similares. A Figura 8.10 apresenta o módulo de gerenciamento desenvolvido para gerenciar as variáveis do Módulo *Reserveds*.

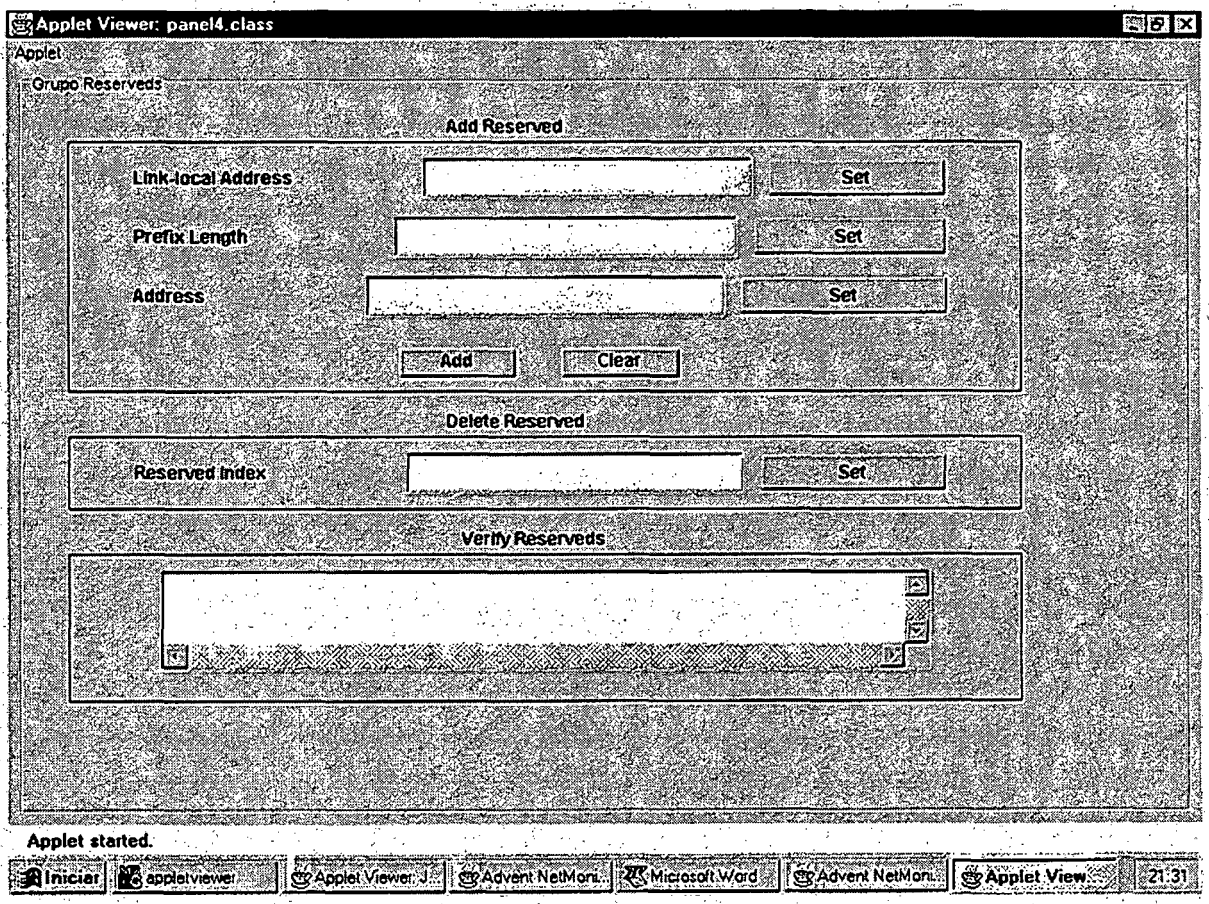


Figura 8. 10 Gerenciamento utilizando Variáveis do Módulo *Reserveds*

Módulo 3 – Grupo *Bindings*

Este módulo faz a verificação do conteúdo das variáveis pertencentes a uma tabela. A Figura 8.11 apresenta o módulo de gerenciamento desenvolvido para gerenciar as variáveis do Módulo *Bindings*.

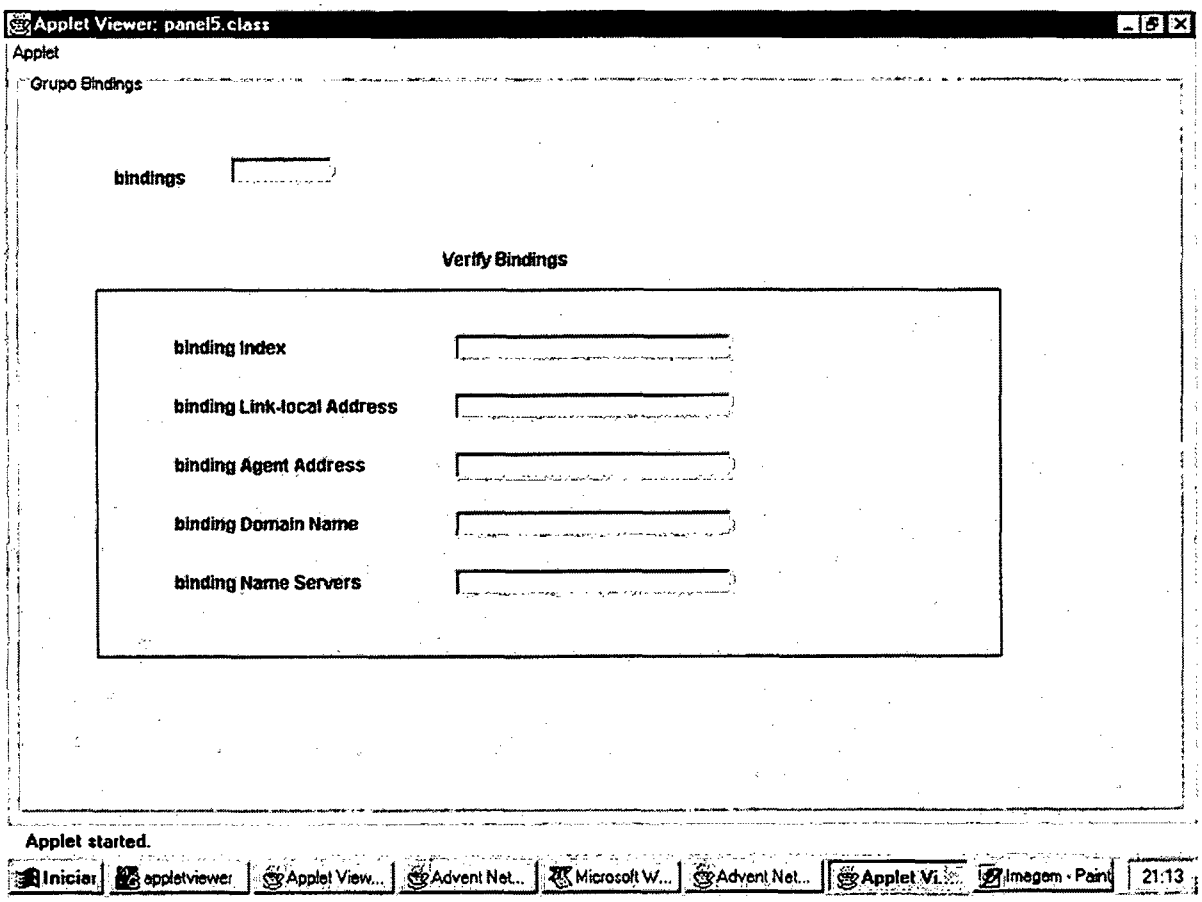


Figura 8. 11 Gerenciamento utilizando Variáveis do Módulo *Bindings*

Módulo 4 - *Notifications*

Este módulo foi construído para demonstrar o recebimento de notificações enviadas pelo agente SNMP ao gerente SNMP. A Figura 8.12 apresenta o módulo de recebimento de notificações; neste exemplo é demonstrado o envio de uma notificação ao gerente após a detecção pelo agente que o processo DHCPv6 foi inicializado.

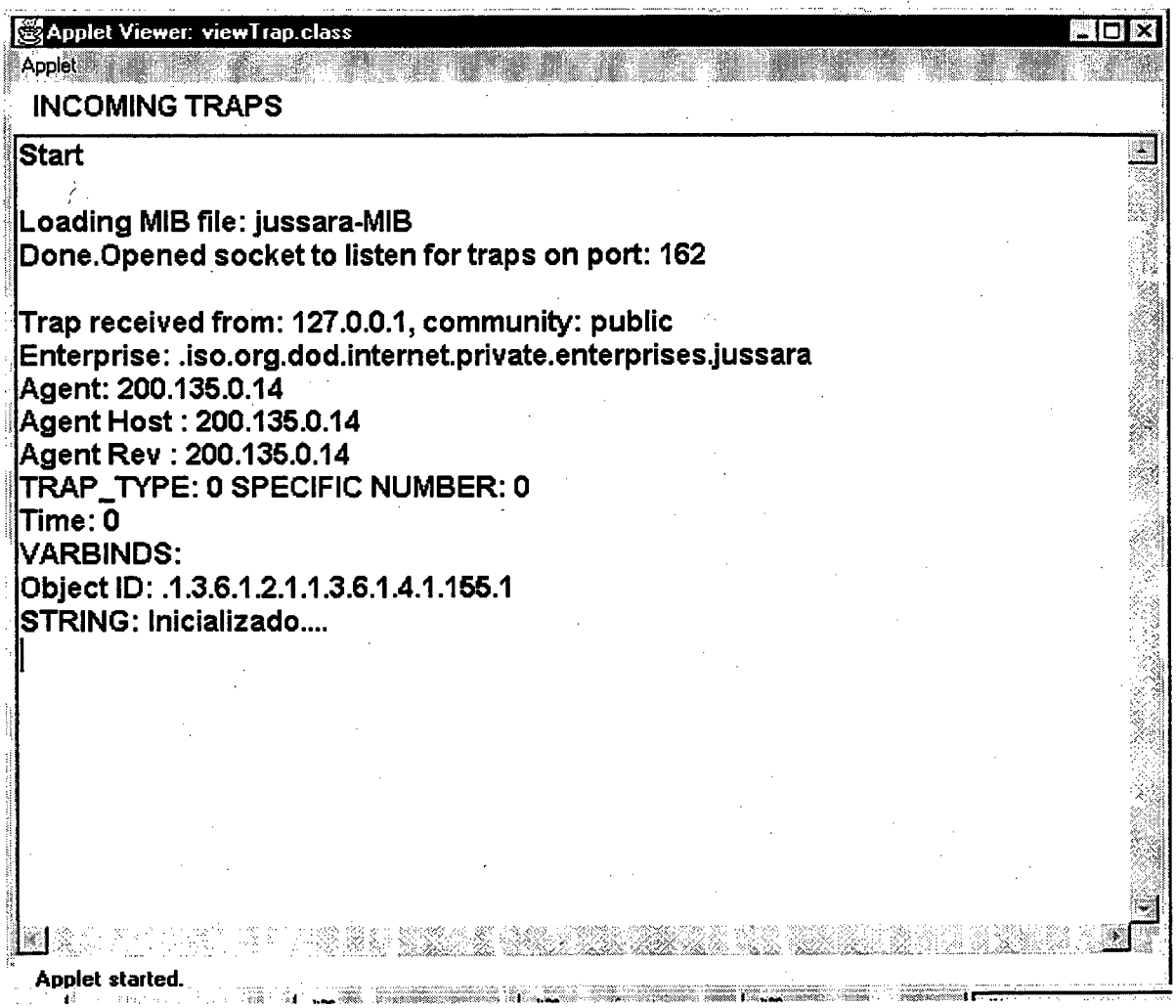


Figura 8. 12 Notificação enviada pelo Agente DHCPv6 informando a inicialização do Processo DHCPv6

8.6 Aspectos de Segurança

Certas informações de gerenciamento definidas nestas MIB podem ser consideradas sensíveis a alguns ambientes de rede. Dessa forma, a autenticação de pedidos SNMP e acesso controlado à informação de gerenciamento devem ser empregadas em tais ambientes.

9 CONCLUSÃO

Segundo as estatísticas [WIZ98], a *Internet* tem crescido significativamente desde o seu surgimento. Seu projeto inicial não previu este nível de expansão. O endereço IP com 32 bits pode representar um número significativo de *hosts*. Entretanto, possivelmente no início do próximo século a capacidade de endereçamento da *Internet* estará esgotada. Mas não é só isso que vem atormentando a comunidade *Internet*. Outros problemas, tem causado muitos danos a sua integridade. A explosão e a instabilidade nas tabelas de roteamento, são causadoras principalmente de perda de datagramas e super utilização dos recursos (CPU, memória, etc.). Muitas vezes a informação de roteamento é redundante e patológica [LAB97]. A fragmentação de datagramas e ausência de métodos nativos de autenticação e privacidade fragiliza a *Internet*, não garantindo a seus usuários a plena confiança no seu uso. Considerando as aplicações multimídia, os recursos na *Internet* devem ser utilizados e fornecidos de forma diferente para cada tipo de aplicação. A qualidade do serviço é ponto chave neste processo evolutivo, pois a *Internet* cresceu, mas este crescimento se deu às custas de muito trabalho. Cada *host* deve ser configurado e testado para se interconectar. Autoconfiguração e possibilidade de mudança de um *host* para uma outra subrede sem maiores dificuldades é uma necessidade eminente atualmente.

Com o surgimento do IPv6 vislumbra-se a solução imediata para os problemas mais críticos, principalmente os relacionados ao endereçamento. Além disso, possibilita-se a existência de novos conceitos e tecnologias que tinham sua potencialidade limitada com o IPv4.

Mundialmente o IPv6 é uma realidade. Ele foi formalmente adotado em agosto de 1997, durante um encontro da IETF em Munique. Várias empresas já implementam ou implementarão o IPv6 nativamente em seus sistemas operacionais. Entretanto, funções de gerenciamento utilizando o IPv6 estão em fase inicial de desenvolvimento, estando o caminho aberto a inúmeras pesquisas.

Nesse sentido, o trabalho proposto teve três propósitos. Em primeiro lugar, elaborar um texto didático que apresentasse a evolução do protocolo IP, passando

por seus principais conceitos, com ênfase especial nos métodos de autoconfiguração propostos para o IPv6. A pesquisa bibliográfica realizada foi extensa, e voltada, principalmente ao entendimento de alguns padrões contidos em RFCs (relacionados principalmente ao IPv6 e gerência de redes SNMP).

Outro objetivo, foi apresentar alguns exemplos práticos de autoconfiguração *stateless*, que servirão como base para novos estudos nesta área, pois bibliografia com este conteúdo ainda não foram disponibilizadas.

O terceiro e principal objetivo, resultante dos estudos teóricos e práticos foi a construção das MIBs que permitem o controle e a monitoração dos serviços de autoconfiguração em redes com o IPv6. Esta fase envolveu o conhecimento aprofundado dos mecanismos de endereçamento e autoconfiguração adotados pelo IPv6.

9.1 Desafios Encontrados

Dentre os desafios encontrados pode-se citar:

- estudo e compreensão de uma extensa lista de padrões *Internet*, fundamentais para a compreensão do protocolo IPv6, bem como, estudo e compreensão dos mecanismos de gerência SNMP, principalmente em sua versão 2;
- estudo detalhado para a compreensão dos métodos de autoconfiguração propostos pelo IPv6;
- construção de um ambiente IPv6, uma vez que não havia nenhuma rede IPv6 disponível na UFSC, onde grande parte deste trabalho foi idealizado. Uma série de tarefas tiveram que ser realizadas, como por exemplo atualização de sistema operacional e configuração de todos os equipamentos envolvidos na construção do ambiente prático;
- reconhecimento do protocolo IP, seu comportamento, características de configuração, etc. Uma série de testes foram realizadas para possibilitar a compreensão dos conceitos teóricos aprendidos;

- estudo e compreensão de MIBs até então desenvolvidas, principalmente as do grupo IPv6;
- implementação de MIBs de autoconfiguração: não foi uma tarefa simples ajustar as necessidade de gerenciamento a uma estrutura de gerenciamento adequada, pois existem limitações no padrão SNMP que impedem a representação fiel de uma configuração remota.

9.2 Resultados Obtidos

Vários foram os resultados obtidos com a realização deste trabalho. Dentre os principais pode-se citar:

- Conhecimento teórico e prático das características do IPv6, com ênfase no aprendizado das técnicas de autoconfiguração. Esta experiência já está permitindo a troca de informações com centros de pesquisa que estão estudando o IPv6;
- A composição de um texto teórico revelando aspectos práticos permite que o conteúdo seja utilizado como referência bibliográfica, principalmente de língua portuguesa;
- Motivou a conexão da UFSC ao *6bone*, sendo a única universidade brasileira a ter acesso a este backbone além do núcleo principal da RNP (Rede Nacional de Pesquisa), a quem estamos diretamente conectados. Este trabalho pioneiro, com certeza é fator estimulante para a demais universidades;
- Certeza da importância dos mecanismos de autoconfiguração e gerência;
- As MIBs desenvolvidas são essenciais para agilizar o controle dos mecanismos de autoconfiguração, principalmente quando se imagina a amplitude que as redes que compõe a *Internet* terão;
- Proposta à IETF para adoção das MIBs implementadas;

- Criação de um modelo de agente de informação de atividade de *hosts*. Este modelo surgiu a partir da elaboração de MIB DHCPv6;
- Proposição de uma extensão DHCPv6 para enviar parâmetros de configuração SNMP.

As seções apresentadas a seguir mostrarão em detalhes alguns dos resultados citados acima.

9.2.1 MIB *IPV6-STATELESS*

Pela simplicidade do método de autoconfiguração *stateless,* a MIB *IPV6-STATELESS* desenvolvida para o seu gerenciamento também é simples e eficaz. A escolha das variáveis foram baseadas na mensagem *Neighbor Discovery Router Advertisement* através da qual os *hosts* receberão os parâmetros de configuração do roteador e de algumas variáveis de configuração específicas para o roteador, como por exemplo se o roteador estará apto para enviar *Router Advertisements.*

A implementação da MIB *IPV6-STATELESS* se mostrou eficaz nos testes de avaliação executados, não apresentando nenhuma inconsistência. Através desta MIB será possível a verificação e/ou a configuração de mensagens *Router Advertisement* que serão enviadas através das diversas interfaces de rede que um roteador apresentar.

Acredita-se que esta implementação da MIB *IPV6-STATELESS* está em condições de ser enviada para os grupos de avaliação da IETF para se tornar uma MIB padrão.

9.2.2 MIB DHCPv6

Construir a MIB DHCPv6 não foi um tarefa simples, além da compreensão do método, que é muito mais complexo que o método de autoconfiguração *stateless,* o ajuste de suas necessidades de gerenciamento a uma estrutura de gerenciamento SNMP

exigiu um certo grau de esforço. Os próximos parágrafos conterão algumas considerações a respeito dos grupos desenvolvidos na MIB DHCPv6.

O grupo *General* permite que se verifique se o processo DHCPv6 está ativo ou não, oferecendo a possibilidade de sua ativação (*start*) ou desativação (*stop*), dependendo do administrador de rede a atitude a ser tomada. Este grupo representa a atividade essencial de um processo que é sua ativação ou desativação. Os testes feitos com este grupo foram positivos e nenhuma dificuldade associada foi encontrada.

O grupo *Variables* foi construído para que o administrador tenha a possibilidade de definir os valores da variáveis de tempo que serão utilizadas no processo de transmissão e retransmissão de mensagens DHCPv6 enviadas pelo servidor aos seus clientes. Estas variáveis são definidas explicitamente pelo método DHCPv6, sendo o seu emprego totalmente justificado. Da mesma forma que o grupo *General* os testes de avaliação, ou seja, verificação através de mensagem SNMP *Get* e alteração do conteúdo através de mensagens SNMP *Set* tiveram os resultados esperados.

O grupo *Bindings* representa a estrutura de dados que o servidor DHCPv6 possui, através do qual são mantidos os dados de configuração que foram repassados para seus clientes. Como a estrutura é uma tabela, onde cada entrada representa os dados de um cliente, optou-se na utilização de um variável colunar (tabela) para a sua representação. Esta tabela permitirá que a verificação do conteúdo da tabela *bindings* que se encontra no servidor DHCPv6, bem como através de um campo adicional introduzido permitirá que se verifique a atividade ou não de um determinado cliente (ver Seção 9.2.3). Através deste grupo o administrador terá condições de constatar se a política de configuração está sendo realmente adotada pelos *hosts* que se autoconfiguraram. Comandos SNMP *Get* e *Set*, aplicados a uma tabela, exigidos pela implementação foram testados e aprovados.

O grupo *Statistics* foi introduzido para fornecer ao administrador valores estatísticos relacionados às mensagens DHCPv6 trocadas entre o servidor e seus clientes. Estas variáveis darão condições ao administrador de verificar entre outras coisas, o grau de atividade do servidor DHCPv6, indicando se está sobrecarregado, necessitando que seu trabalho seja compartilhado por outros servidores. O grupo *Statistics* também permite que erros na troca de mensagens sejam detectados, diagnosticando assim determinados erros de configuração. Além destas funções, possibilita que notificações sejam enviadas durante

uma tentativa de acesso a rede sem autorização prévia. A construção e os testes realizados com este grupo não apresentaram maiores dificuldades, com exceção da determinação correta das variáveis que o compõe.

Os grupos *Reserveds*, *Excludeds*, e *Nets* possuem uma estrutura muito similar, ou seja, são formados por três subgrupos, sendo que o primeiro subgrupo é responsável pela adição de elementos, o segundo subgrupo pela verificação do conteúdo das variáveis e finalmente o terceiro grupo responsável pela remoção de elementos. Todos os três grupos, *Reserveds*, *Excludes* e *Nets*, apresentam atividade críticas, onde a implementação do agente deverá ser a mais criteriosa possível, principalmente, nas atividade de adição e exclusão. Os valores ditados só devem ser atribuídos mediante rigorosa validação. Os testes realizados comprovaram a validade da implementação, mas somente através de uma rigorosa avaliação do conteúdo de adição e/ou remoção é que se mostram eficientes e confiáveis. Estes grupos são de grande importância, pois agilizam o processo de configuração dos servidores DHCPv6. Entretanto, por apresentarem atividades críticas, a implementação do subgrupos responsáveis pela adição e remoção não é considerada obrigatória. As estrutura destes grupos não estão explicitamente ditadas pelo protocolo DHCPv6, mas através do estudo e da compreensão deste protocolo fica claro a necessidade desta estrutura. Através do grupo *Reserveds* será feita a adição, verificação e exclusão de endereços que devem ser atribuídos especificamente a um determinado *host*. O grupo *Excludeds* permite que o administrador gerencie aqueles endereços que não devem ser atribuídos dinamicamente, tendo seu emprego reservado para outra finalidade. O grupo *Nets* tem um a finalidade de permitir a definição do parâmetros de configuração genéricos que serão repassados pelo agente DHCPv6. Para cada agente, mais especificamente para cada interface de um agente, será possibilitado que uma estrutura de configuração específica seja montada. As variáveis envolvidas são aquelas definidas pelas extensões DHCPv6 propostas.

O grupo *Notifications* permite que o administrador seja notificado que atividades ou situações críticas estão sendo realizadas. A escolha das variáveis foi criteriosa para não haver uma superutilização de mensagens de notificação, podendo causar uma sobrecarga na utilização dos recursos da rede, pois possivelmente a todo momento uma notificação estaria sendo enviada ao gerente. As notificações implementadas estão

relacionadas mais diretamente com os aspectos de segurança, tentando manter o administrador ciente de configurações realizadas e, principalmente, de tentativas de acesso não autorizado. Uma notificação mais específica foi definida funcionará como agente de informação de atividade (ver Seção 9.2.3). A implementação do agente que trata de notificações foi um pouco mais trabalhosa, entretanto se mostrou totalmente eficaz.

Sem dúvida, existe a possibilidade do refinamento desta MIB, principalmente, relacionado a sua estrutura. Com relação as variáveis empregadas, acredita-se que são as essenciais, pois se mostraram totalmente eficientes nas suas atribuições.

A MIB *IPV6-STATELESS* e a MIB DHCPv6 serão enviadas como propostas à IETF, entretanto os grupos críticos (*Reserveds, Excludeds* e *Nets)* serão apresentados mais simplificados, os quais apenas as funções de verificação estarão incluídas. A versão final só será apresentada posteriormente.

9.2.3 MODELO DE AGENTE DE INFORMAÇÃO DE ATIVIDADE DE *HOSTS*

A monitoração da atividade de *hosts,* ou seja, a verificação se o *host* está conectado ao *link* e com TCP/IP ativo, é sem dúvida uma das funções mais utilizadas e úteis no gerenciamento de redes. Este tipo de monitoração entretanto, utiliza muito recurso da rede, pois periodicamente o gerente tem que emitir mensagens verificadoras, que muitas vezes trafegam por todos os *links* de uma rede, gerando uma sobrecarga de tráfego generalizada.

Na MIB DHCPv6 foi introduzida uma variável que verifica a atividade dos *hosts* que receberam a configuração a partir do método *stateful.* A implementação desta permite que seja emitido ao gerente uma notificação quando for identificado pelo agente, que um *host* teve seu estado de atividade alterado. Como provavelmente existirão vários servidores DHCPv6 em uma rede, isto proporcionará um monitoração distribuída. O gerente não terá mais a função de monitoração de atividade de *hosts*; esta atividade será distribuída entre os servidores DHCPv6 de uma rede. Além disso, a monitoração estará limitada a um ambiente restrito, não envolvendo toda rede, pois, os servidores DHCPv6 servirão uma pequena área e, exatamente, nesta área é que se realizará a monitoração, não afetando toda rede. Outra vantagem associada a este método é que diminuirá a sobrecarga

de atividades do sistema de gerenciamento principalmente em sua porção gerente. Normalmente as ferramentas de gerenciamento são construídas em ambientes gráficos que super utilizam os recursos das máquinas que os implementam. A porção agente por sua vez tem a sua tarefa (monitoração da atividade) distribuída entre os vários servidores DHCPv6.

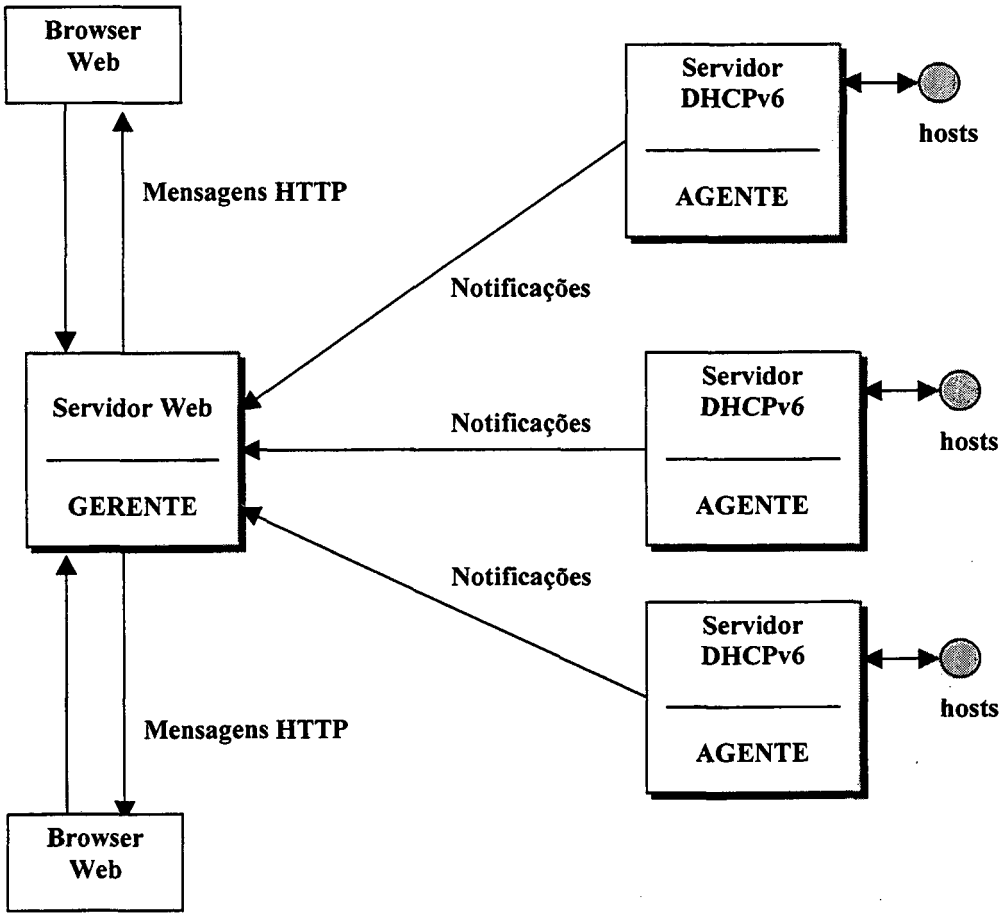


Figura 9. 1 Modelo de Gerenciamento de Atividade de *Hosts*

A figura 9.1 ilustra o modelo de gerenciamento de atividade resultante dos estudos realizados neste projeto que além de envolver conceitos de gerenciamento SNMP e IPv6, utiliza os novos modelos de gerenciamento que utilizam as tecnologias *Web* e Java (ver capítulo 7) como ferramentas de gerenciamento. Neste modelo, tem-se quatro entidades: o *browser Web*(1), através do qual o administrador receberá as notificações das

atividades. O *browser Web* interagirá diretamente com o servidor HTTP(2), através de mensagens HTTP, que por sua vez terá uma relação direta com o gerente SNMP(2) que estará sendo processando na mesma máquina, ou seja a visualização de entrada e saída de comando SNMP é feita através de páginas Web fornecidas pelo servidor HTTP. Ao agente/Servidor DHCPv6(3) cabe verificar a atividade dos *hosts*(4) e enviar notificações ao gerente diante de uma mudança de estado. É importante salientar que a figura 9.1 apenas ilustra o gerenciamento de atividade de *hosts*. Entretanto, este modelo se ajusta as outras funções implementadas nas MIBs propostas.

9.2.4 EXTENSÃO DHCPV6 SNMP

Este assunto está sendo objeto de estudo. Ela proporcionará ao clientes DHCPv6 receberem configurações de gerenciamento. Esta extensão está sendo desenvolvida baseada nos padrões de configuração e segurança do SNMPv3 [BLU97]. A figura 9.2 mostra o formato da extensão que está sendo desenvolvida. Essencialmente ela é baseada na MIB USM (*User-based Security Model*) e os campos que a compõe ainda estão em fase de estudo e não serão descritos neste projeto.

Type	Length
UserEngineID	UserName
UserSecurityName	UserCloneFrom
UserAuthProtocol	UserAuthKeyChange
UserOwnAuthKeyChange	UserPrivProtocol
UserPrivKeyChange	UserOwnPrivKeyChange
UserPublic	User StorageType
UserStatus	

Figura 9. 2 Protótipo de Extensão DHCPv6 para a Configuração de Parâmetros SNMP

9.3 Perspectivas Futuras

O conteúdo deste trabalho permitiu abordar as perspectivas de pesquisas futuras relacionadas a três tópicos principais, apresentados a seguir:

IPv6

Os estudos sobre o IPv6 estão em fase inicial, o campo para pesquisas neste assunto é vasto. Como pôde ser verificado o protocolo introduziu uma série de novidades que precisam ter seus desempenhos e funcionalidades. Devidamente testados e compreendidos. A pesquisa realizada neste trabalho abordou apenas uma porção de todo o universo IPv6. Na própria área de autoconfiguração o protocolo SLP (*Service Location Protocol*) e os mecanismos de atualização dinâmica do DNS (*DNS Update*) merecem um estudo aprofundado.

Dentre os pontos principais para desenvolvimento de novos projetos pode-se citar, pesquisas de análise de desempenho de redes IPv6 utilizando os recursos de segurança adotados, estudos na área de qualidade de serviço, mobilidade de *hosts*, desempenho de redes IPv6 sobre as tecnologias de redes como por exemplo: ATM, FDDI e Ethernet.

Gerenciamento SNMP

O protocolo SNMP se mostrou eficiente na sua tarefa de gerenciamento por muitos anos. Entretanto, a sua estrutura não comporta mais as necessidades de segurança e configuração apresentadas nos dias atuais. A versão 3 do protocolo SNMP, surgiu para resolver estes problemas. Porém, ela é relativamente nova, necessitando de ajustes e testes que a validem totalmente.

IPv6 *versus* Gerenciamento SNMP

O gerenciamento do protocolo IPv6 é compatível com o SNMPv1 e SNMPv2 (possivelmente será compatível com o SNMPv3). Entretanto, os grupos da MIB *Internet* devem ser avaliados para determinar se devem ser reescritos para se ajustarem ao novo protocolo. Além disso, os novos protocolos que surgiram ou que estão surgindo possivelmente necessitarão ter uma MIB associada. Outra possibilidade é o desenvolvimento de MIBs privadas associadas a equipamentos específicos que implementam o protocolo IPv6.

9.4 Conclusão Final

Este trabalho contribuiu em dois aspectos principais:

- apresenta no seu conteúdo teórico conceitos e características do protocolo IPv6 que certamente contribuirão para o aperfeiçoamento teórico de pesquisadores, administradores de rede, etc.;
- As MIBs propostas mostram-se adequadas para que sejam empregadas, além disso, servirão de modelo para a construção de outras MIBs que estarão associadas ao IPv6 e que utilizem o protocolo SNMPv2;

Acredita-se que este trabalho venha a contribuir para que o IPv6 e gerência de redes se tornem cada vez mais objeto de estudo da comunidade científica, uma vez que são temas de extrema importância que devem ser continuamente aperfeiçoados.

ANEXO 1

- MIB IPV6-STATELESS -

```
IPV6-STATELESS DEFINITIONS ::= BEGIN
IMPORTS

      iso                                                    FROM RFC1213-MIB
      MODULE-IDENTITY, OBJECT-TYPE, NOTIFICATION-TYPE, mib-2,
      Unsigned32,    Integer32                               FROM SNMPv2-SMI
      DisplayString                                          FROM SNMv2-TC
      MODULE-COMPLIANCE, OBJECT-GROUP                        FROM SNMPv2-CONF
      Ipv6IfIndex, Ipv6AddressPrefix
      FROM IPV6-TC;

org   OBJECT IDENTIFIER  ::=  {  iso  3  }

dod   OBJECT IDENTIFIER  ::=  {  org  6  }

internet    OBJECT IDENTIFIER  ::=  {  dod  1  }

directory   OBJECT IDENTIFIER  ::=  {  internet  1  }

mgmt  OBJECT IDENTIFIER  ::=  {  internet  2  }

experimental      OBJECT IDENTIFIER  ::=  {  internet  3  }

private     OBJECT IDENTIFIER  ::=  {  internet  4  }

snmpV2      OBJECT IDENTIFIER  ::=  {  internet  6  }

mib-2 OBJECT IDENTIFIER  ::=  {  mgmt  1  }

enterprises OBJECT IDENTIFIER  ::=  {  private  1  }

snmpDomains OBJECT IDENTIFIER  ::=  {  snmpV2  1  }

snmpProxys  OBJECT IDENTIFIER  ::=  {  snmpV2  2  }

snmpModules OBJECT IDENTIFIER  ::=  {  snmpV2  3  }

ufsc  OBJECT IDENTIFIER  ::=  {  enterprises  100  }

ipv6StatelessMIB  MODULE-IDENTITY
      LAST-UPDATED      "2147483647 Z"
      ORGANIZATION      "Universidade Federal de Santa Catarina"
      CONTACT-INFO      "Jussara Maria Bozzano

Postal: Nucleo de Processamento de Dados
Campus Universitario
Trindade
Florianopolis, SC, Brazil
88040 -900
```

```
Tel: +55-331-7839
Fax: +55-331-9766

E-mail: jussara@npd.ufsc.br"
      DESCRIPTION
            "Modulo MIB para roteadores que implementam o servico
de autoconfiguracao stateless - IPv6."
  ::=  {  ufsc  2  }

ipv6StatelessMIBObjects OBJECT IDENTIFIER  ::=  {  ipv6StatelessMIB  1  }

--
--
-- Modulo General
--
--

ipv6SlessInterfaces OBJECT-TYPE
      SYNTAX      Unsigned32
      MAX-MAX-ACCESS    read-only
      STATUS      current

--    AGENTCLAUSE
--           "
--           FILE-COMMAND:
--           FILE-NAME: mibs/datafiles/data/interface.txt
--           FIELD-SEPARATOR: ' '
--           FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE

           ::=  { ipv6StatelessMIBObjects  1 }


ipv6IfSlessTable  OBJECT-TYPE
      SYNTAX      SEQUENCE  OF  Ipv6IfSlessEntry
      MAX-MAX-ACCESS    not-accessible
      STATUS      current
--    AGENTCLAUSE
--           "
--           FILE-COMMAND:
--           FILE-NAME: mibs/datafiles/data/dataTable
--           FIELD-SEPARATOR: ' '
--           FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  { ipv6StatelessMIBObjects  2 }

ipv6IfSlessEntry  OBJECT-TYPE
      SYNTAX      Ipv6IfSlessEntry
      MAX-MAX-ACCESS    not-accessible
      STATUS      current
      INDEX {  ipv6IfSlessIndex, ipv6IfSlessNdAdRcTm, ipv6IfSlessNdRtAdI,
ipv6IfSlessHl, ipv6IfSlessNdRtAdLt, ipv6IfSlessNdNsI,
ipv6IfSlessNdSupRtAd, ipv6IfSlessNdPrefixAd, ipv6IfSlessNdPrefixLength,
ipv6IfSlessManAddConfFlag, ipv6IfSlessOtherStatefulConfFlag,
ipv6IfSlessMtu  }
      ::=  { ipv6IfSlessTable 1 }
```

```
Ipv6IfSlessEntry  ::=  SEQUENCE {
      iv6IfSlessIndex  Ipv6IfIndex,
      ipv6IfSlessNdAdRcTm  Integer32,
      ipv6IfSlessNdRtAdI  Integer32,
      ipv6IfSlessNdHl  INTEGER,
      ipv6IfSlessNdRtAdLt  Integer32,
      ipv6IfSlessNdNsI  Integer32,
      ipv6IfSlessNdSupRtAd  INTEGER,
      ipv6IfSlessNdPrefixAd  Ipv6AddressPrefix,
      ipv6IfSlessNdPrefixLength  INTEGER,
      ipv6IfSlessManAddConfFlag  INTEGER,
      ipv6IfSlessOtherStatefulConfFlag  INTEGER,
      ipv6IfSlessMtu  Unsigned32
      }

iv6IfSlessIndex  OBJECT-TYPE
      SYNTAX      Ipv6IfIndex
      MAX-ACCESS  read-only
      STATUS      current

      ::=  {  ipv6IfSlessEntry  1  }

ipv6IfSlessNdAdRcTm    OBJECT-TYPE
      SYNTAX      Integer32
      MAX-ACCESS  read-write
      STATUS      current

      ::=  {  ipv6IfSlessEntry  2  }

ipv6IfSlessNdRtAdI     OBJECT-TYPE
      SYNTAX      Integer32
      MAX-ACCESS  read-write
      STATUS      current

      ::=  {  ipv6IfSlessEntry  3  }

ipv6IfSlessNdHl  OBJECT-TYPE
      SYNTAX      INTEGER (0..255)
      MAX-ACCESS  read-write
      STATUS      current
      ::=  {  ipv6IfSlessEntry  4  }

ipv6IfSlessNdRtAdLt    OBJECT-TYPE
      SYNTAX      Integer32
      MAX-ACCESS  read-write
      STATUS      current
      ::=  {  ipv6IfSlessEntry  5  }

ipv6IfSlessNdNsI  OBJECT-TYPE
      SYNTAX      Integer32
      MAX-ACCESS  read-write
      STATUS      current
      ::=  {  ipv6IfSlessEntry  6  }

ipv6IfSlessNdSupRtAd   OBJECT-TYPE
      SYNTAX      INTEGER { off ( 1 ) , on ( 2 ) }
```

```
      MAX-ACCESS  read-write
      STATUS      current
      ::=  {  ipv6IfSlessEntry  7  }

ipv6IfSlessNdPrefixAd  OBJECT-TYPE
      SYNTAX      Ipv6AddressPrefix
      MAX-ACCESS  read-write
      STATUS      current
      ::=  {  ipv6IfSlessEntry  8  }

ipv6IfSlessNdPrefixLength     OBJECT-TYPE
      SYNTAX      INTEGER (0 .. 128)
      MAX-ACCESS  read-write
      STATUS      current
      ::=  {  ipv6IfSlessEntry  9  }



ipv6IfSlessManAddConfFlag     OBJECT-TYPE
      SYNTAX      INTEGER { off ( 1 ) , on ( 2 ) }
      MAX-ACCESS  read-write
      STATUS      current
      ::=  {  ipv6IfSlessEntry  10  }

ipv6IfSlessOtherStatefulConfFlag   OBJECT-TYPE
      SYNTAX      Integer32 { off ( 1 ) , on ( 2 ) }
      MAX-ACCESS  read-write
      STATUS      current
      ::=  {  ipv6IfSlessEntry  11  }

ipv6IfSlessMtu    OBJECT-TYPE
      SYNTAX      Unsigned32
      MAX-ACCESS  read-write
      STATUS      current
      ::=  {  ipv6IfSlessEntry  12  }

--
--
-- CONFORMANCE INFORMATION
--
--

ipv6StatelessConformance      OBJECT IDENTIFIER  ::=  {  ipv6StatelessMIB
2  }

ipv6StatelessCompliances      OBJECT IDENTIFIER  ::=  {
ipv6StatelessConformance  1  }

ipv6StatelessGroups     OBJECT IDENTIFIER  ::=  {
ipv6StatelessConformance  2  }
--
--
-- COMPLIANCE STATEMENTS
--
--
dhcpv6Compliance MODULE-COMPLIANCE
      STATUS current
```

```
      DESCRIPTION " "

      MODULE
            MANDATORY-GROUPS  { ipv6StatelessGeneralGroup
                                }

            OBJECT ipv6IfSlessNdAdRcTm
                  MIN-ACCESS read-only
                  DESCRIPTION " "

            OBJECT ipv6IfSlessNdRtAdI
                  MIN-ACCESS read-only
                  DESCRIPTION " "

            OBJECT ipv6IfSlessNdHl
                  MIN-ACCESS read-only
                  DESCRIPTION " "

            OBJECT ipv6IfSlessNdRtAdLt
                  MIN-ACCESS read-only
                  DESCRIPTION " "

            OBJECT ipv6IfSlessNdNsI
                  MIN-ACCESS read-only
                  DESCRIPTION " "

            OBJECT ipv6IfSlessNdSupRtAd
                  MIN-ACCESS read-only
                  DESCRIPTION " "

            OBJECT ipv6IfSlessNdPrefixAd
                  MIN-ACCESS read-only
                  DESCRIPTION " "

            OBJECT ipv6IfSlessNdPrefixLength
                  MIN-ACCESS read-only
                  DESCRIPTION " "


            OBJECT ipv6IfSlessManAddConfFlag
                  MIN-ACCESS read-only
                  DESCRIPTION " "

            OBJECT      ipv6IfSlessOtherStatefulConfFlag
                  MIN-ACCESS read-only
                  DESCRIPTION " "

            OBJECT ipv6IfSlessMtu
                  MIN-ACCESS read-only
                  DESCRIPTION " "


  ::= { ipv6StatelessCompliances 1 }

ipv6StatelessGeneralGroup OBJECT-GROUP
      OBJECTS {ipv6IfSlessIndex, ipv6IfSlessNdAdRcTm, ipv6IfSlessNdRtAdI,
ipv6IfSlessHl, ipv6IfSlessNdRtAdLt, ipv6IfSlessNdNsI,
```

```
ipv6IfSlessNdSupRtAd, ipv6IfSlessNdPrefixAd, ipv6IfSlessNdPrefixLength,
ipv6IfSlessManAddConfFlag, ipv6IfSlessOtherStatefulConfFlag,
ipv6IfSlessMtu }
STATUS      current
      DESCRIPTION " "
::= { ipv6StatelessGroups 1 }


END
```

ANEXO 2

- MIB DHCPv6 -

```
DHCPv6-MIB DEFINITIONS ::= BEGIN
IMPORTS

      MODULE-IDENTITY, OBJECT-TYPE, NOTIFICATION-TYPE, mib-2, Counter32,
      Unsigned32, Integer32                                  FROM SNMPv2-SMI
      DisplayString, PhysAddress, TimeStamp, TEXTUAL-CONVENTION
                                                              FROM SNMPv2-TC
      MODULE-COMPLIANCE, OBJECT-GROUP, NOTIFICATION-GROUP
                                                            FROM SNMPv2-CONF
      Ipv6IfIndex,Ipv6Address, Ipv6AddressPrefix, Ipv6AddressIfIdentifier
                                                               FROM IPV6-TC;
--
--
-- TEXTUAL-CONVENTION
--
--
Index ::= TEXTUAL-CONVENTION
      DISPLAY-HINT "d"
      STATUS current
      SYNTAX Integer32 (1 .. 2147483647)

org   OBJECT IDENTIFIER  ::=  {  iso  3  }

dod   OBJECT IDENTIFIER  ::=  {  org  6  }

internet    OBJECT IDENTIFIER  ::=  {  dod  1  }

mgmt  OBJECT IDENTIFIER  ::=  {  internet  2  }

experimental      OBJECT IDENTIFIER  ::=  {  internet  3  }

private     OBJECT IDENTIFIER  ::=  {  internet  4  }

mib-2 OBJECT IDENTIFIER  ::=  {  mgmt  1  }

enterprises OBJECT IDENTIFIER  ::=  {  private 1  }

ufsc  OBJECT IDENTIFIER  ::=  {  enterprises  100  }

dhcpv6MIB   OBJECT IDENTIFIER  ::=  {  ufsc  1  }

-- MODULE-IDENTITY
-- LastUpdated
-- Z
-- OrgName
-- Universidade Federal de Santa Catarina
-- ContactInfo
-- Jussara Maria Bozzano
-- Postal: Nucleo de Processamento de Dados
-- Campus Universitario
```

```
-- Trindade
-- Florianopolis , SC
-- Tel: +55 048 331 7839
-- Fax: +55 048 331 9766
-- E-mail: jussara@npd.ufsc.br
-- Descr
-- The MIB module for entities implementing the DHCPv6
-- protocol .
dhcpv6MIBObjects  OBJECT IDENTIFIER  ::=  {  dhcpv6MIB  1  }

--
--
-- Modulo General
--
--

upTime      OBJECT-TYPE
      SYNTAX      TimeTicks
      MAX-ACCESS  read-only
      STATUS      current

--    AGENTCLAUSE
--           "
--           FILE-COMMAND:
--           FILE-NAME: mibs/datafiles/upTime.txt
--           FIELD-SEPARATOR: ' '
--           FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  dhcpv6MIBObjects  1  }

operation   OBJECT-TYPE
      SYNTAX      INTEGER { start ( 1 ) , stop ( 2 ) , refresh ( 3 ) }
      MAX-ACCESS  read-write
      STATUS      current

--    AGENTCLAUSE
--           "
--          WRITE-COMMAND:
mibs/datafiles/operation.exe $value
--          WRITE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  dhcpv6MIBObjects  2  }

message     OBJECT-TYPE
      SYNTAX      DisplayString
      MAX-ACCESS  read-only
      STATUS      current

      DESCRIPTION
            "Por onde as mensagens de atividades e erros serao anunciadas
por este servidor DHCP"

--    AGENTCLAUSE
--           "
--           FILE-COMMAND:
--           FILE-NAME: mibs/datafiles/message.txt
--           FIELD-SEPARATOR: ' '
```

```
--            FILE-COMMAND-TIMEOUT: 5000"
--      END AGENTCLAUSE
       ::=  {  dhcpv6MIBObjects  3  }


varSMinAD   OBJECT-TYPE
       SYNTAX       Integer32
       MAX-ACCESS   read-write
       STATUS       current


--AGENTCLAUSE
--"
-- FILE-COMMAND:
-- FILE-NAME: mibs/datafiles/varSMinAD.txt
-- FILE-COMMAND-TIMEOUT: 5000 "
-- END AGENTCLAUSE

       REFERENCE
             ""

--     AGENTCLAUSE
--            "
--            FILE-COMMAND:
--            FILE-NAME: mibs/datafiles/varSMinAD.txt
--            FIELD-SEPARATOR: ' '
--            FILE-COMMAND-TIMEOUT: 5000"
--      END AGENTCLAUSE
       ::=  {  dhcpv6MIBObjects  4  }

varSMaxAD   OBJECT-TYPE
       SYNTAX       Integer32
       MAX-ACCESS   read-write
       STATUS       current

       REFERENCE
             ""

--     AGENTCLAUSE
--            "
--            FILE-COMMAND:
--            FILE-NAME: mibs/datafiles/varSMaxAD.txt
--            FIELD-SEPARATOR: ' '
--            FILE-COMMAND-TIMEOUT: 5000"
--      END AGENTCLAUSE
       ::=  {  dhcpv6MIBObjects  5  }

varRecMsgTimeout  OBJECT-TYPE
       SYNTAX       Integer32
       MAX-ACCESS   read-write
       STATUS       current


       REFERENCE
             ""

--     AGENTCLAUSE
```

```
--            "
--            FILE-COMMAND:
--            FILE-NAME: mibs/datafiles/varRecMsgTimeout.txt
--            FIELD-SEPARATOR: ' '
--            FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  dhcpv6MIBObjects  6  }

varRecMsgMinRet   OBJECT-TYPE
      SYNTAX      Integer32
      MAX-ACCESS  read-write
      STATUS      current


--    AGENTCLAUSE
--            "
--            FILE-COMMAND:
--            FILE-NAME: mibs/datafiles/varRecMsgRet.txt
--            FIELD-SEPARATOR: ' '
--            FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  dhcpv6MIBObjects  7  }

varRecMsgRetI     OBJECT-TYPE
      SYNTAX      Integer32
      MAX-ACCESS  read-write
      STATUS      current


--    AGENTCLAUSE
--            "
--            FILE-COMMAND:
--            FILE-NAME: mibs/datafiles/varRecMsgRetI.txt
--            FIELD-SEPARATOR: ' '
--            FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  dhcpv6MIBObjects  8  }

varXidTimeout     OBJECT-TYPE
      SYNTAX      Integer32
      MAX-ACCESS  read-write
      STATUS      current

--    AGENTCLAUSE
--            "
--            FILE-COMMAND:
--            FILE-NAME: mibs/datafiles/varXidTimeout.txt
--            FIELD-SEPARATOR: ' '
--            FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  dhcpv6MIBObjects  9  }

--
--
-- Modulo Reserveds
--
--
```

```
addReservedPrefixLength OBJECT-TYPE
      SYNTAX      Integer32 (0 .. 128)
      MAX-ACCESS  read-write
      STATUS      current


--    AGENTCLAUSE
--           "
--           FILE-COMMAND:
--           FILE-NAME: mibs/datafiles/addReserved.txt
--           FIELD-SEPARATOR: ' '
--           FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  dhcpv6MIBObjects  10  }

addReservedLinkLocalAddress  OBJECT-TYPE
      SYNTAX      Ipv6Address
      MAX-ACCESS  read-write
      STATUS      current

--    AGENTCLAUSE
--           "
--           FILE-COMMAND:
--           FILE-NAME: mibs/datafiles/addReservedLynkLocalAddress.txt
--           FIELD-SEPARATOR: ' '
--           FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  dhcpv6MIBObjects  11  }

addReservedAddress      OBJECT-TYPE
      SYNTAX      Ipv6Address
      MAX-ACCESS  read-write
      STATUS      current

--    AGENTCLAUSE
--           "
--           FILE-COMMAND:
--           FILE-NAME: mibs/datafiles/addReservedAddress.txt
--           FIELD-SEPARATOR: ' '
--           FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  dhcpv6MIBObjects  12  }

addReservedAct    OBJECT-TYPE
      SYNTAX      INTEGER { add ( 1 ), clear ( 2 ) }
      MAX-ACCESS  read-write
      STATUS      current

      DESCRIPTION
            ""

      REFERENCE
            ""

--    AGENTCLAUSE
--           "
```

```
--          WRITE-COMMAND:
mibs/datafiles/addReservedAct.exe $value
--          WRITE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  dhcpv6MIBObjects  13  }

reserveds   OBJECT-TYPE
      SYNTAX      INTEGER
      MAX-ACCESS  read-only
      STATUS      current

--    AGENTCLAUSE
--          "
--          FILE-COMMAND:
--          FILE-NAME: mibs/datafiles/reserveds.txt
--          FIELD-SEPARATOR: ' '
--          FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  dhcpv6MIBObjects  14  }


reservedTable     OBJECT-TYPE
      SYNTAX      SEQUENCE  OF  ReservedEntry
      MAX-ACCESS  not-accessible
      STATUS      current
--    AGENTCLAUSE
--          "
--          FILE-COMMAND:
--          FILE-NAME: mibs/datafiles/netTable.txt
--          FIELD-SEPARATOR: ' '
--          FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  { dhcpv6MIBObjects  15 }

reservedEntry     OBJECT-TYPE
      SYNTAX      ReservedEntry
      MAX-ACCESS  not-accessible
      STATUS      current
      INDEX {  reservedPrefixLength  }
      ::=  { reservedTable 1 }

ReservedEntry  ::=  SEQUENCE {
      reservedIndex  Index,
      reservedPrefixLength  INTEGER,
      reservedLynkLocal  Ipv6Address,
      reservedAddress  Ipv6Address
      }
reservedIndex     OBJECT-TYPE
      SYNTAX      Index
      MAX-ACCESS  read-only
      STATUS      current

      ::=  {  reservedEntry  1  }

reservedPrefixLength    OBJECT-TYPE
      SYNTAX      INTEGER ( 0 .. 128 )
      MAX-ACCESS  read-only
```

```
      STATUS      current

      ::=  {  reservedEntry  2  }

reservedLinkLocal OBJECT-TYPE
      SYNTAX      Ipv6Address
      MAX-ACCESS  read-only
      STATUS      current

      ::=  {  reservedEntry  3  }

reservedAddress   OBJECT-TYPE
      SYNTAX      Ipv6Address
      MAX-ACCESS  read-only
      STATUS      current

      ::=  {  reservedEntry  4  }

deleteReserved    OBJECT-TYPE
      SYNTAX      Index
      MAX-ACCESS  read-write
      STATUS      current

--    AGENTCLAUSE
--          "
--          WRITE-COMMAND:
mibs/datafiles/deleteReserved $value
--          WRITE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  dhcpv6MIBObjects  16  }

--
--
-- Modulo Excludeds
--
--

addExcludedPrefixLength OBJECT-TYPE
      SYNTAX      INTEGER ( 0 .. 128 )
      MAX-ACCESS  read-write
      STATUS      current

--    AGENTCLAUSE
--          "
--          FILE-COMMAND:
--          FILE-NAME: mibs/datafiles/addExcludedPrefixLength.txt
--          FIELD-SEPARATOR: ' '
--          FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  dhcpv6MIBObjects  17  }

addExcludedRangeBegin  OBJECT-TYPE
      SYNTAX      Ipv6Address
      MAX-ACCESS  read-write
      STATUS      current

--    AGENTCLAUSE
```

```
--            "
--            FILE-COMMAND:
--            FILE-NAME: mibs/datafiles/addExcludeRangeBegin.txt
--            FIELD-SEPARATOR: ' '
--            FILE-COMMAND-TIMEOUT: 5000"
--      END AGENTCLAUSE
      ::=  {  dhcpv6MIBObjects  18  }

addExcludedRangeEnd     OBJECT-TYPE
      SYNTAX      Ipv6Address
      MAX-ACCESS  read-write
      STATUS      current

--    AGENTCLAUSE
--            "
--            FILE-COMMAND:
--            FILE-NAME: mibs/datafiles/addExcludeRangeEnd.txt
--            FIELD-SEPARATOR: ' '
--            FILE-COMMAND-TIMEOUT: 5000"
--      END AGENTCLAUSE
      ::=  {  dhcpv6MIBObjects  19  }

addExcludedAct    OBJECT-TYPE
      SYNTAX      INTEGER { add ( 1 ), clear ( 2 ) }
      MAX-ACCESS  read-write
      STATUS      current

      DESCRIPTION
            ""

      REFERENCE
            ""

--    AGENTCLAUSE
--            "
--          WRITE-COMMAND:
mibs/datafiles/addExcludeAct.exe $value
--          WRITE-COMMAND-TIMEOUT: 5000"
--      END AGENTCLAUSE
      ::=  {  dhcpv6MIBObjects  20  }

excludeds   OBJECT-TYPE
      SYNTAX      INTEGER
      MAX-ACCESS  read-only
      STATUS      current

--    AGENTCLAUSE
--            "
--            FILE-COMMAND:
--            FILE-NAME: mibs/datafiles/excludeds.txt
--            FIELD-SEPARATOR: ' '
--            FILE-COMMAND-TIMEOUT: 5000"
--      END AGENTCLAUSE
      ::=  {  dhcpv6MIBObjects  21  }


excludedTable     OBJECT-TYPE
```

```
        SYNTAX      SEQUENCE  OF  ExcludeEntry
        MAX-ACCESS  not-accessible
        STATUS      current
--      AGENTCLAUSE
--            "
--            FILE-COMMAND:
--            FILE-NAME: mibs/datafiles/excludedTable.txt
--            FIELD-SEPARATOR: ' '
--            FILE-COMMAND-TIMEOUT: 5000"
--       END AGENTCLAUSE
        ::=  { dhcpv6MIBObjects  22 }

excludedEntry     OBJECT-TYPE
        SYNTAX      ExcludeEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        INDEX {  excludedIndex  }
        ::=  { excludeTable 1 }

ExcludedEntry  ::=  SEQUENCE {
        excludedIndex  Index,
        excludedPrefixLengthAddress  INTEGER,
        excludedRangeBegin  Ipv6Address,
        excludedRangeEnd  Ipv6Address
        }
excludedIndex     OBJECT-TYPE
        SYNTAX      INTEGER
        ACCESS      read-only
        STATUS      current

        ::=  {  excludeEntry  1  }

excludedPrefixLengthAddress  OBJECT-TYPE
        SYNTAX      INTEGER ( 0 .. 128 )
        ACCESS      read-only
        STATUS      current

        ::=  {  excludeEntry  2  }

excludeRangeBegin OBJECT-TYPE
        SYNTAX      Ipv6Address
        ACCESS      read-only
        STATUS      current

        ::=  {  excludeEntry  3  }

excludeRangeEnd   OBJECT-TYPE
        SYNTAX      Ipv6Address
        ACCESS      read-only
        STATUS      current

        ::=  {  excludeEntry  4  }

deleteExclude     OBJECT-TYPE
        SYNTAX      Index
        MAX-ACCESS  read-write
        STATUS      current
```

```
--     AGENTCLAUSE
--             "
--           WRITE-COMMAND:
mibs/datafiles/deleteExclude.exe $value
--           WRITE-COMMAND-TIMEOUT: 5000"
--      END AGENTCLAUSE
       ::=  {  dhcpv6MIBObjects  23  }

--
--
-- Modulo Bindings
--
--

bindings    OBJECT-TYPE
       SYNTAX       INTEGER
       MAX-ACCESS   read-only
       STATUS       current

--     AGENTCLAUSE
--             "
--             FILE-COMMAND:
--             FILE-NAME: mibs/datafiles/bindings.txt
--             FIELD-SEPARATOR: ' '
--             FILE-COMMAND-TIMEOUT: 5000"
--      END AGENTCLAUSE
       ::=  {  dhcpv6MIBObjects  24  }


bindingsTable      OBJECT-TYPE
       SYNTAX       SEQUENCE  OF  BindingsEntry
       MAX-ACCESS   not-accessible
       STATUS       current
--     AGENTCLAUSE
--             "
--             FILE-COMMAND:
--             FILE-NAME: mibs/datafiles/bindingTable.txt
--             FIELD-SEPARATOR: ' '
--             FILE-COMMAND-TIMEOUT: 5000"
--      END AGENTCLAUSE
       ::=  { dhcpv6MIBObjects  25 }

bindingsEntry      OBJECT-TYPE
       SYNTAX       BindingsEntry
       MAX-ACCESS   not-accessible
       STATUS       current
       INDEX {  bindingIndex  }
       ::=  { bindingsTable 1 }

BindingsEntry  ::=  SEQUENCE {
       bindingIndex  Index,
       bindingLynkLocalAddress  Ipv6Address,
       bindingAgentAddressPrefix  Ipv6AddressPrefix,
       bindingPreferLifeTime  INTEGER,
       bindingValidLifeTime  INTEGER,
       bindingTimeOffSet  INTEGER,
```

```
      bindingPosixTime  DisplayString,
      bindingNameServers  DisplayString,
      bindingDomainName  DisplayString,
      bindingDirAgent  DisplayString,
      bindingDirAgentScopeList  DisplayString,
      bindingServiceScope  DisplayString,
      bindingNtpServers  DisplayString,
      bindingNisDomainName  DisplayString,
      bindingNisServers  DisplayString,
      bindingNisPlusDomainName  DisplayString,
      bindingNisPlusServers  DisplayString,
      bindingKeepAliveInterval  INTEGER,
      bindingMaxDhcpv6Message  INTEGER,
      bindingClientServerAuthSpi  INTEGER,
      bindingClientServerAuthRepProtec  TimeStamp,
      bindingStatus  INTEGER,
      bindingActivitie  INTEGER
      }
bindingIndex      OBJECT-TYPE
      SYNTAX      Index
      MAX-ACCESS  read-only
      STATUS      current

      ::=  {  bindingsEntry  1  }

bindingLynkLocalAddress OBJECT-TYPE
      SYNTAX      Ipv6Address
      MAX-ACCESS  read-only
      STATUS      current

--    AGENTCLAUSE
--          "
--          FILE-COMMAND:
--          FILE-NAME: mibs/datafiles/bindingLynkLocalAddress.txt
--          FIELD-SEPARATOR: ' '
--          FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  bindingsEntry  2  }

bindingAgentAddressPrefix     OBJECT-TYPE
      SYNTAX      Ipv6AddressPrefix
      MAX-ACCESS  read-only
      STATUS      current

--    AGENTCLAUSE
--          "
--          FILE-COMMAND:
--          FILE-NAME: mibs/datafiles/bindingAgentAddressPrefix.txt
--          FIELD-SEPARATOR: ' '
--          FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  bindingsEntry  3  }

bindingPreferLifeTime   OBJECT-TYPE
      SYNTAX      INTEGER
      MAX-ACCESS  read-only
      STATUS      current
```

```
--    AGENTCLAUSE
--           "
--           FILE-COMMAND:
--           FILE-NAME: mibs/datafiles/bindingPreferLifeTime.txt
--           FIELD-SEPARATOR: ' '
--           FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  bindingsEntry  4  }

bindingValidLifeTime    OBJECT-TYPE
      SYNTAX      INTEGER
      MAX-ACCESS  read-only
      STATUS      current

--    AGENTCLAUSE
--           "
--           FILE-COMMAND:
--           FILE-NAME: mibs/datafiles/bindingValidLifeTime.txt
--           FIELD-SEPARATOR: ' '
--           FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  bindingsEntry  5  }

bindingTimeOffSet OBJECT-TYPE
      SYNTAX      INTEGER
      MAX-ACCESS  read-only
      STATUS      current

--    AGENTCLAUSE
--           "
--           FILE-COMMAND:
--           FILE-NAME: mibs/datafiles/bindingTimeOffSete.txt
--           FIELD-SEPARATOR: ' '
--           FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  bindingsEntry  6  }

bindingPosixTime  OBJECT-TYPE
      SYNTAX      DisplayString
      MAX-ACCESS  read-only
      STATUS      current

--    AGENTCLAUSE
--           "
--           FILE-COMMAND:
--           FILE-NAME: mibs/datafiles/bindingPosixTime.txt
--           FIELD-SEPARATOR: ' '
--           FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  bindingsEntry  7  }

bindingNameServers      OBJECT-TYPE
      SYNTAX      DisplayString
      MAX-ACCESS  read-only
      STATUS      current
```

```
--     AGENTCLAUSE
--             "
--             FILE-COMMAND:
--             FILE-NAME: mibs/datafiles/bindingNameServers.txt
--             FIELD-SEPARATOR: ' '
--             FILE-COMMAND-TIMEOUT: 5000"
--      END AGENTCLAUSE
       ::=  {  bindingsEntry  8  }

bindingDomainName OBJECT-TYPE
       SYNTAX       DisplayString
       MAX-ACCESS   read-only
       STATUS       current

--     AGENTCLAUSE
--             "
--             FILE-COMMAND:
--             FILE-NAME: mibs/datafiles/bindingDomainName.txt
--             FIELD-SEPARATOR: ' '
--             FILE-COMMAND-TIMEOUT: 5000"
--      END AGENTCLAUSE
       ::=  {  bindingsEntry  9  }

bindingDirAgent   OBJECT-TYPE
       SYNTAX       DisplayString
       MAX-ACCESS   read-only
       STATUS       current

--     AGENTCLAUSE
--             "
--             FILE-COMMAND:
--             FILE-NAME: mibs/datafiles/bindingDirAgent.txt
--             FIELD-SEPARATOR: ' '
--             FILE-COMMAND-TIMEOUT: 5000"
--      END AGENTCLAUSE
       ::=  {  bindingsEntry  10  }

bindingDirAgentScopeList      OBJECT-TYPE
       SYNTAX       DisplayString
       MAX-ACCESS   read-only
       STATUS       current

--     AGENTCLAUSE
--             "
--             FILE-COMMAND:
--             FILE-NAME: mibs/datafiles/bindingDirAgentScopeList.txt
--             FIELD-SEPARATOR: ' '
--             FILE-COMMAND-TIMEOUT: 5000"
--      END AGENTCLAUSE
       ::=  {  bindingsEntry  11  }

bindingServiceScope     OBJECT-TYPE
       SYNTAX       DisplayString
       MAX-ACCESS   read-only
       STATUS       current

--     AGENTCLAUSE
```

```
--            "
--            FILE-COMMAND:
--            FILE-NAME: mibs/datafiles/bindingServiceScope.txt
--            FIELD-SEPARATOR: ' '
--            FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  bindingsEntry  12  }

bindingNtpServers OBJECT-TYPE
      SYNTAX      DisplayString
      MAX-ACCESS  read-only
      STATUS      current

--    AGENTCLAUSE
--            "
--            FILE-COMMAND:
--            FILE-NAME: mibs/datafiles/bindingNtpServer.txt
--            FIELD-SEPARATOR: ' '
--            FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  bindingsEntry  13  }

bindingNisDomainName    OBJECT-TYPE
      SYNTAX      DisplayString
      MAX-ACCESS  read-only
      STATUS      current

--    AGENTCLAUSE
--            "
--            FILE-COMMAND:
--            FILE-NAME: mibs/datafiles/bindingNisDomainName.txt
--            FIELD-SEPARATOR: ' '
--            FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  bindingsEntry  14  }

bindingNisServers OBJECT-TYPE
      SYNTAX      DisplayString
      MAX-ACCESS  read-only
      STATUS      current

--    AGENTCLAUSE
--            "
--            FILE-COMMAND:
--            FILE-NAME: mibs/datafiles/bindingNisServers.txt
--            FIELD-SEPARATOR: ' '
--            FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  bindingsEntry  15  }

bindingNisPlusDomainName      OBJECT-TYPE
      SYNTAX      DisplayString
      MAX-ACCESS  read-only
      STATUS      current

--    AGENTCLAUSE
--            "
```

```
--            FILE-COMMAND:
--            FILE-NAME: mibs/datafiles/NisPlusDomainName.txt
--            FIELD-SEPARATOR: ' '
--            FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  bindingsEntry  16  }

bindingNisPlusServers   OBJECT-TYPE
      SYNTAX      DisplayString
      MAX-ACCESS  read-only
      STATUS      current

--    AGENTCLAUSE
--            "
--            FILE-COMMAND:
--            FILE-NAME: mibs/datafiles/bindingNisPlusServers.txt
--            FIELD-SEPARATOR: ' '
--            FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  bindingsEntry  17  }

bindingKeepAliveInterval      OBJECT-TYPE
      SYNTAX      INTEGER
      MAX-ACCESS  read-only
      STATUS      current

      DESCRIPTION
            ""

      REFERENCE
            ""

--    AGENTCLAUSE
--            "
--            FILE-COMMAND:
--            FILE-NAME: mibs/datafiles/bindingKeepAliveInterval.txt
--            FIELD-SEPARATOR: ' '
--            FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  bindingsEntry  18  }

bindingMaxDhcpv6Message OBJECT-TYPE
      SYNTAX      INTEGER
      MAX-ACCESS  read-only
      STATUS      current

--    AGENTCLAUSE
--            "
--            FILE-COMMAND:
--            FILE-NAME: mibs/datafiles/bindingMaxDhcpv6Message.txt
--            FIELD-SEPARATOR: ' '
--            FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  bindingsEntry  19  }

bindingClientServerAuthSpi    OBJECT-TYPE
      SYNTAX      INTEGER
```

```
      MAX-ACCESS  read-only
      STATUS      current

--    AGENTCLAUSE
--          "
--          FILE-COMMAND:
--          FILE-NAME: mibs/datafiles/bindingClientServerAuthSpi.txt
--          FIELD-SEPARATOR: ' '
--          FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  bindingsEntry  20  }

bindingClientServerAuthRepProtec   OBJECT-TYPE
      SYNTAX      TimeStamp
      MAX-ACCESS  read-only
      STATUS      current

--    AGENTCLAUSE
--          "
--          FILE-COMMAND:
--                                                                FILE-NAME:
mibs/datafiles/bindingClientServerAuthrepProtec.txt
--          FIELD-SEPARATOR: ' '
--          FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  bindingsEntry  21  }

bindingStatus     OBJECT-TYPE
      SYNTAX      INTEGER
      MAX-ACCESS  read-only
      STATUS      current

--    AGENTCLAUSE
--          "
--          FILE-COMMAND:
--          FILE-NAME: mibs/datafiles/bindingStatus.txt
--          FIELD-SEPARATOR: ' '
--          FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  bindingsEntry  22  }

bindingActivitie  OBJECT-TYPE
      SYNTAX      INTEGER
      MAX-ACCESS  read-only
      STATUS      current

--    AGENTCLAUSE
--          "
--          FILE-COMMAND:
--          FILE-NAME: mibs/datafiles/bindingActivitie.txt
--          FIELD-SEPARATOR: ' '
--          FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  bindingsEntry  23  }

statSolicit OBJECT-TYPE
      SYNTAX      Counter32
```

```
      MAX-ACCESS  read-only
      STATUS      current

--    AGENTCLAUSE
--           "
--           FILE-COMMAND:
--           FILE-NAME: mibs/datafiles/statSolicit.txt
--           FIELD-SEPARATOR: ' '
--           FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  dhcpv6MIBObjects  26  }

statAdvertise     OBJECT-TYPE
      SYNTAX      Counter32
      MAX-ACCESS  read-only
      STATUS      current

--    AGENTCLAUSE
--           "
--           FILE-COMMAND:
--           FILE-NAME: mibs/datafiles/statAdvertise.txt
--           FIELD-SEPARATOR: ' '
--           FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  dhcpv6MIBObjects  27  }

statRequest OBJECT-TYPE
      SYNTAX      Counter32
      MAX-ACCESS  read-only
      STATUS      current

--    AGENTCLAUSE
--           "
--           FILE-COMMAND:
--           FILE-NAME: mibs/datafiles/statRequest.txt
--           FIELD-SEPARATOR: ' '
--           FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  dhcpv6MIBObjects  28  }

statRelease OBJECT-TYPE
      SYNTAX      Counter32
      MAX-ACCESS  read-only
      STATUS      current

--    AGENTCLAUSE
--           "
--           FILE-COMMAND:
--           FILE-NAME: mibs/datafiles/statRelease.txt
--           FIELD-SEPARATOR: ' '
--           FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  dhcpv6MIBObjects  29  }

statReconfigure   OBJECT-TYPE
      SYNTAX      Counter32
      MAX-ACCESS  read-only
```

```
      STATUS      current

--    AGENTCLAUSE
--          "
--          FILE-COMMAND:
--          FILE-NAME: mibs/datafiles/statReconfigure.txt
--          FIELD-SEPARATOR: ' '
--          FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  dhcpv6MIBObjects  30  }

statReplyUnspError      OBJECT-TYPE
      SYNTAX      Counter32
      MAX-ACCESS  read-only
      STATUS      current

--    AGENTCLAUSE
--          "
--          FILE-COMMAND:
--          FILE-NAME: mibs/datafiles/statReplyUnspError.txt
--          FIELD-SEPARATOR: ' '
--          FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  dhcpv6MIBObjects  31  }

statReplyAuthError      OBJECT-TYPE
      SYNTAX      Counter32
      MAX-ACCESS  read-only
      STATUS      current

--    AGENTCLAUSE
--          "
--          FILE-COMMAND:
--          FILE-NAME: mibs/datafiles/statReplyAuthError.txt
--          FIELD-SEPARATOR: ' '
--          FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  dhcpv6MIBObjects  32  }

statReplyFormedError    OBJECT-TYPE
      SYNTAX      Counter32
      MAX-ACCESS  read-only
      STATUS      current

--    AGENTCLAUSE
--          "
--          FILE-COMMAND:
--          FILE-NAME: mibs/datafiles/statReplyFormedError.txt
--          FIELD-SEPARATOR: ' '
--          FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  dhcpv6MIBObjects  33  }

statReplyResourcesError OBJECT-TYPE
      SYNTAX      Counter32
      MAX-ACCESS  read-only
      STATUS      current
```

```
--     AGENTCLAUSE
--             "
--             FILE-COMMAND:
--             FILE-NAME: mibs/datafiles/statReplyResourcesError.txt
--             FIELD-SEPARATOR: ' '
--             FILE-COMMAND-TIMEOUT: 5000"
--      END AGENTCLAUSE
       ::=  {  dhcpv6MIBObjects  34  }

statReplyClientRecError OBJECT-TYPE
       SYNTAX       Counter32
       MAX-ACCESS   read-only
       STATUS       current

--     AGENTCLAUSE
--             "
--             FILE-COMMAND:
--             FILE-NAME: mibs/datafiles/statReplyClientRecError.txt
--             FIELD-SEPARATOR: ' '
--             FILE-COMMAND-TIMEOUT: 5000"
--      END AGENTCLAUSE
       ::=  {  dhcpv6MIBObjects  35  }

statReplyInvalidClientIpError OBJECT-TYPE
       SYNTAX       Counter32
       MAX-ACCESS   read-only
       STATUS       current

--     AGENTCLAUSE
--             "
--             FILE-COMMAND:
--             FILE-NAME: mibs/datafiles/statReplyInvalidClientIpError.txt
--             FIELD-SEPARATOR: ' '
--             FILE-COMMAND-TIMEOUT: 5000"
--      END AGENTCLAUSE
       ::=  {  dhcpv6MIBObjects  36  }

statCharSetError  OBJECT-TYPE
       SYNTAX       Counter32
       MAX-ACCESS   read-only
       STATUS       current

--     AGENTCLAUSE
--             "
--             FILE-COMMAND:
--             FILE-NAME: mibs/datafiles/statChatSetError.txt
--             FIELD-SEPARATOR: ' '
--             FILE-COMMAND-TIMEOUT: 5000"
--      END AGENTCLAUSE
       ::=  {  dhcpv6MIBObjects  37  }

activitie   OBJECT-TYPE
       SYNTAX       INTEGER {
       MAX-ACCESS   read-only
       STATUS       current
```

```
--     AGENTCLAUSE
--             "
--             FILE-COMMAND:
--             FILE-NAME: mibs/datafiles/activitie.txt
--             FIELD-SEPARATOR: ' '
--             FILE-COMMAND-TIMEOUT: 5000"
--      END AGENTCLAUSE
       ::=  {  dhcpv6MIBObjects  38  }

--
--
-- NET GROUP
--
addNetDesc  OBJECT-TYPE
       SYNTAX       DisplayString
       MAX-ACCESS   read-write
       STATUS       current
       ::=  {  dhcpv6MIBObjects  39  }

addNetPrefix
       SYNTAX         Ipv6AddressPrefix
       MAX-ACCESS   read-write
       STATUS       current
       ::=  {  dhcpv6MIBObjects  40  }


addNetRangeBegin     OBJECT-TYPE
       SYNTAX         Ipv6AddressPrefix
       MAX-ACCESS   read-write
       STATUS       current
       ::=  {  dhcpv6MIBObjects  41  }



addNetRangeEnd      OBJECT-TYPE
       SYNTAX        Ipv6AddressPrefix
       MAX-ACCESS   read-write
       STATUS       current
       ::=  {  dhcpv6MIBObjects  42  }



addNetLeasedTime     OBJECT-TYPE
       SYNTAX         Integer32
       MAX-ACCESS     read-write
       STATUS         current
       ::=  {  dhcpv6MIBObjects  43  }

addNetTimeOffSet     OBJECT-TYPE
       SYNTAX         Integer32
       MAX-ACCESS     read-write
       STATUS         current
       ::=  {  dhcpv6MIBObjects  44  }


addNetTimePosix      OBJECT-TYPE
       SYNTAX         Integer32
```

```
      MAX-ACCESS    read-write
      STATUS        current
      ::=  {  dhcpv6MIBObjects  45  }

addNetNameServers         OBJECT-TYPE
      SYNTAX                  DisplayString
      MAX-ACCESS    read-write
      STATUS        current
      ::=  {  dhcpv6MIBObjects  46  }


addNetDomainName    OBJECT-TYPE
      SYNTAX                  DisplayString
      MAX-ACCESS    read-write
      STATUS        current
      ::=  {  dhcpv6MIBObjects  47  }

addNetDirAgent      OBJECT-TYPE
      SYNTAX                  DisplayString
      MAX-ACCESS    read-write
      STATUS        current
      ::=  {  dhcpv6MIBObjects  48  }

addNetDirAgentScopeList   OBJECT-TYPE
      SYNTAX                  DisplayString
      MAX-ACCESS    read-write
      STATUS        current
      ::=  {  dhcpv6MIBObjects  49  }

addNetServiceScope        OBJECT-TYPE
      SYNTAX                  DisplayString
      MAX-ACCESS    read-write
      STATUS        current
      ::=  {  dhcpv6MIBObjects  50  }

addNetNtpServers    OBJECT-TYPE
      SYNTAX                  DisplayString
      MAX-ACCESS    read-write
      STATUS        current
      ::=  {  dhcpv6MIBObjects  51  }

addNetNisDomainName       OBJECT-TYPE
      SYNTAX                  DisplayString
      MAX-ACCESS    read-write
      STATUS        current
      ::=  {  dhcpv6MIBObjects  44  }

addNetNisServers    OBJECT-TYPE
      SYNTAX                  DisplayString
      MAX-ACCESS    read-write
      STATUS        current
      ::=  {  dhcpv6MIBObjects  45  }


addNetNisPlusDomainName   OBJECT-TYPE
      SYNTAX                  DisplayString
      MAX-ACCESS    read-write
```

```
      STATUS        current
      ::=  {  dhcpv6MIBObjects  46  }


addNetNisPlusServers      OBJECT-TYPE
      SYNTAX                  DisplayString
      MAX-ACCESS    read-write
      STATUS        current
      ::=  {  dhcpv6MIBObjects  47  }


addNetKeepAliveInterval   OBJECT-TYPE
      SYNTAX                  DisplayString
      MAX-ACCESS    read-write
      STATUS        current
      ::=  {  dhcpv6MIBObjects  48  }



addNetMaxDhcpv6Msg        OBJECT-TYPE
      SYNTAX                  DisplayString
      MAX-ACCESS    read-write
      STATUS        current
      ::=  {  dhcpv6MIBObjects  49  }


addNetReconfigure   OBJECT-TYPE
      SYNTAX                  Integer32
      MAX-ACCESS    read-write
      STATUS        current
      ::=  {  dhcpv6MIBObjects  50  }



addNetAct     OBJECT-TYPE
      SYNTAX                  Integer32
      MAX-ACCESS    read-write
      STATUS        current
      ::=  {  dhcpv6MIBObjects  51  }


deleteNet   OBJECT-TYPE
      SYNTAX      Index
      MAX-ACCESS  read-write
      STATUS      current

--    AGENTCLAUSE
--          "
--          WRITE-COMMAND:
mibs/datafiles/deleteReserved $value
--          WRITE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  dhcpv6MIBObjects  52  }
```

```
nets  OBJECT-TYPE
      SYNTAX      Integer32
      MAX-ACCESS  read-only
      STATUS      current

      DESCRIPTION
            "Numero de redes passiveis de serem configuradas por este
servidor DHCP"

--    AGENTCLAUSE
--           "
--           FILE-COMMAND:
--           FILE-NAME: mibs/datafiles/networks.txt
--           FIELD-SEPARATOR: ' '
--           FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  dhcpv6MIBObjects  53  }


netTable    OBJECT-TYPE
      SYNTAX      SEQUENCE  OF  NetEntry
      MAX-ACCESS  not-accessible
      STATUS      current
--    AGENTCLAUSE
--           "
--           FILE-COMMAND:
--           FILE-NAME: mibs/datafiles/netTable.txt
--           FIELD-SEPARATOR: ' '
--           FILE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  { dhcpv6MIBObjects  54 }

netEntry    OBJECT-TYPE
      SYNTAX      NetEntry
      MAX-ACCESS  not-accessible
      STATUS      current
      INDEX {  netIndex  }
      ::=  { netTable 1 }

NetEntry  ::=  SEQUENCE {
      netIndex  Index,
      netDesc  DisplayString,
      netPrefix  Ipv6AddressPrefix,
      netRangeBegin  Ipv6AddressPrefix,
      netRangeEnd  Ipv6AddressPrefix,
      netLeasedTime  Integer32,
      netLastChange  TimeStamp,
      netTimeOffSet  Integer32,
      netTimePosix  Integer32,
      netNameServers  DisplayString,
      netDomainName  DisplayString,
      netDirAgent  DisplayString,
      netDirAgentScopeList  DisplayString,
      netServiceScope  DisplayString,
      netNtpServers  DisplayString,
      netNisDomainName  DisplayString,
      netNisServers  DisplayString,
```

```
      netNisPlusDomainName  DisplayString,
      netNisPlusServers  DisplayString,
      netKeepAliveInterval  DisplayString,
      netMaxDhcpv6Msg  DisplayString,
      netAdmStatus  Integer32,
      netReconfigure  Integer32,
      netAct      Integer32
      }

netIndex    OBJECT-TYPE
      SYNTAX      Index
      MAX-ACCESS  read-only
      STATUS      current

      ::=  {  netEntry  1  }

netDesc     OBJECT-TYPE
      SYNTAX      DisplayString
      MAX-ACCESS  read-write
      STATUS      current

      ::=  {  netEntry  2  }

netPrefix   OBJECT-TYPE
      SYNTAX      Ipv6AddressPrefix
      MAX-ACCESS  read-write
      STATUS      current

--    AGENTCLAUSE
--          "
--          WRITE-COMMAND:     /mibs/datafiles/netPrefix.exe $value
--          WRITE-COMMAND-TIMEOUT: 5000"
--     END AGENTCLAUSE
      ::=  {  netEntry  3  }

netRangeBegin     OBJECT-TYPE
      SYNTAX      Ipv6AddressPrefix
      MAX-ACCESS  read-only
      STATUS      current

      ::=  {  netEntry  4  }

netRangeEnd OBJECT-TYPE
      SYNTAX      Ipv6AddressPrefix
      MAX-ACCESS  read-write
      STATUS      current

      ::=  {  netEntry  5  }

netLeasedTime     OBJECT-TYPE
      SYNTAX      Integer32
      MAX-ACCESS  read-write
      STATUS      mandatory

      ::=  {  netEntry  6  }

netLastChange     OBJECT-TYPE
```

```
      SYNTAX      TimeStamp
      MAX-ACCESS  read-only
      STATUS      current

      ::=  {  netEntry  7  }

netTimeOffSet     OBJECT-TYPE
      SYNTAX      Integer32
      MAX-ACCESS  read-write
      STATUS      current

      ::=  {  netEntry  8  }

netTimePosix      OBJECT-TYPE
      SYNTAX      Integer32
      MAX-ACCESS  read-write
      STATUS      current

      ::=  {  netEntry  9  }

netNameServers    OBJECT-TYPE
      SYNTAX      DisplayString
      MAX-ACCESS  read-write
      STATUS      current

      ::=  {  netEntry  10  }

netDomainName     OBJECT-TYPE
      SYNTAX      DisplayString
      MAX-ACCESS  read-write
      STATUS      current

      ::=  {  netEntry  11  }

netDirAgent       OBJECT-TYPE
      SYNTAX      DisplayString
      MAX-ACCESS  read-write
      STATUS      current

      ::=  {  netEntry  12  }

netDirAgentScopeList   OBJECT-TYPE
      SYNTAX      DisplayString
      ACCESS      read-write
      STATUS      current

      ::=  {  netEntry  13  }

netServiceScope   OBJECT-TYPE
      SYNTAX      DisplayString
      MAX-ACCESS  read-write
      STATUS      mandatory

      ::=  {  netEntry  14  }

netNtpServers     OBJECT-TYPE
      SYNTAX      DisplayString
```

```
      MAX-ACCESS  read-write
      STATUS      current

      ::=  {  netEntry  15  }

netNisDomainName  OBJECT-TYPE
      SYNTAX      DisplayString
      MAX-ACCESS  read-write
      STATUS      current

      ::=  {  netEntry  16  }

netNisServers     OBJECT-TYPE
      SYNTAX      DisplayString
      MAX-ACCESS  read-write
      STATUS      mandatory

      ::=  {  netEntry  17  }

netNisPlusDomainName    OBJECT-TYPE
      SYNTAX      DisplayString
      MAX-ACCESS  read-write
      STATUS      current

      ::=  {  netEntry  18  }

netNisPlusServers OBJECT-TYPE
      SYNTAX      DisplayString
      MAX-ACCESS  read-write
      STATUS      current

      ::=  {  netEntry  19  }

netKeepAliveInterval    OBJECT-TYPE
      SYNTAX      DisplayString
      MAX-ACCESS  read-write
      STATUS      current

      ::=  {  netEntry  20  }

netMaxDhcpv6Msg   OBJECT-TYPE
      SYNTAX      DisplayString
      MAX-ACCESS  read-write
      STATUS      current

      ::=  {  netEntry  21  }

netAdmStatus      OBJECT-TYPE
      SYNTAX      Integer32
      MAX-ACCESS  read-write
      STATUS      current

      DESCRIPTION
            ""

      REFERENCE
            ""
```

```
--     AGENTCLAUSE
--             "
--           WRITE-COMMAND: mibs/datafiles/netAdmStatus.exe $value
--           WRITE-COMMAND-TIMEOUT: 5000"
--      END AGENTCLAUSE
       ::=  {  netEntry  22  }

netReconfigure     OBJECT-TYPE
       SYNTAX      Integer32
       MAX-ACCESS  read-write
       STATUS      current

       DESCRIPTION
             ""

       REFERENCE
             ""

--     AGENTCLAUSE
--             "
--           WRITE-COMMAND: mibs/datafiles/netReconfigure $value
--           WRITE-COMMAND-TIMEOUT: 5000"
--      END AGENTCLAUSE
       ::=  {  netEntry  23  }


--
--
--  NOTIFICATIONS
--
--

dhcpv6Notifications     OBJECT IDENTIFIER  ::=  {  dhcpv6MIB  2  }

dhcpv6NotificationPrefix      OBJECT      IDENTIFIER          ::=          {
dhcpv6Notifications  0  }


dhcpv6StateChange NOTIFICATION-TYPE
       OBJECTS      { operation  }
       STATUS       current
       DESCRIPTION "null"
  ::=  {  dhcpv6NotificationPrefix  1  }

dhcpv6AddReservedAct    NOTIFICATION-TYPE
       OBJECTS      { addReservedLinkLocal , addReservedAddress  }
       STATUS       current
       DESCRIPTION "null"
  ::=  {  dhcpv6NotificationPrefix  2  }

dhcpv6DelReserved NOTIFICATION-TYPE
       OBJECTS      { reservedAddress , deleteReserved  }
       STATUS       current
       DESCRIPTION "null"
  ::=  {  dhcpv6NotificationPrefix  3  }
```

```
dhcpv6AddExclude  NOTIFICATION-TYPE
      OBJECTS     { excludeRangeBegin , excludeRangeEnd  }
      STATUS      current
      DESCRIPTION "null"
  ::=  {  dhcpv6NotificationPrefix  4  }

dhcpv6DelExclude  NOTIFICATION-TYPE
      OBJECTS     { excludeRangeBegin , excludeRangeEnd , deleteExclude
}
      STATUS      current
      DESCRIPTION "null"
  ::=  {  dhcpv6NotificationPrefix  5  }

dhcpv6ReservedStatus    NOTIFICATION-TYPE
      OBJECTS     { reservedAddress , reservedLinkLocal , reservedStatus
}
      STATUS      current
      DESCRIPTION "null"
  ::=  {  dhcpv6NotificationPrefix  6  }

dhcpv6ExcludeStatus     NOTIFICATION-TYPE
      OBJECTS     { excludeRangeBegin , excludeRangeEnd , excludeStatus
}
      STATUS      current
      DESCRIPTION "null"
  ::=  {  dhcpv6NotificationPrefix  7  }

dhcpv6BindingActivitieChange  NOTIFICATION-TYPE
      OBJECTS     {     bindingAgentPrefix     ,     bindingAddress     ,
bindingActivitie  }
      STATUS      current
      DESCRIPTION "null"
  ::=  {  dhcpv6NotificationPrefix  8  }

dhcpv6StatReplyAuthError      NOTIFICATION-TYPE
      OBJECTS     { statReplyAuthError  }
      STATUS      current
      DESCRIPTION "null"
  ::=  {  dhcpv6NotificationPrefix  9  }
--
--
-- CONFORMANCE INFORMATION
--
--

dhcpv6Conformance OBJECT IDENTIFIER  ::=  {  dhcpv6MIB  3  }

dhcpv6Compliances OBJECT IDENTIFIER  ::=  {  dhcpv6Conformance  1  }

dhcpv6Groups      OBJECT IDENTIFIER  ::=  {  dhcpv6Conformance  2  }
--
--
-- COMPLIANCE STATEMENTS
--
--
dhcpv6Compliance MODULE-COMPLIANCE
      STATUS current
```

```
DESCRIPTION " "

MODULE
      MANDATORY-GROUPS  { dhcpv6GeneralGroup,
                          dhcpv6NotificationGroup }
      OBJECT varSMinAD
            MIN-ACCESS read-only
            DESCRIPTION " "

      OBJECT varSMaxAD
            MIN-ACCESS read-only
            DESCRIPTION " "

      OBJECT varRecMsgTimeout
            MIN-ACCESS read-only
            DESCRIPTION " "

      OBJECT varRecMsgMinRet
            MIN-ACCESS read-only
            DESCRIPTION " "

      OBJECT varRecMsgRetI
            MIN-ACCESS read-only
            DESCRIPTION " "

      OBJECT varXidtTimeout
            MIN-ACCESS read-only
            DESCRIPTION " "

      OBJECT operation
            MIN-ACCESS read-only
            DESCRIPTION " "

      OBJECT reservedPrefixLength
            MIN-ACCESS read-only
            DESCRIPTION " "

      OBJECT reservedLinkLocal
            MIN-ACCESS read-only
            DESCRIPTION " "

      OBJECT reservedAddress
            MIN-ACCESS read-only
            DESCRIPTION " "

      OBJECT excludePrefixLength
            MIN-ACCESS read-only
            DESCRIPTION " "

      OBJECT exludeRangeBegin
            MIN-ACCESS read-only
            DESCRIPTION " "

      OBJECT excludeRangeEnd
            MIN-ACCESS read-only
            DESCRIPTION " "
```

```
  ::= { dhcpv6Compliances 1 }

dhcpv6GeneralGroup OBJECT-GROUP
      OBJECTS {
                  uptime,
                  varSMinAD,
                  varSMaxAD,
                  varRecMsgTimeout,
                  varRecMsgMinRet,
                  varRecMsgRetI,
                  varXidtTimeout,
                  operation,
                  reserveds,
                  reservedIndex,
                  reservedPrefixLength,
                  reservedLinkLocal,
                  reservedAddress,
                  reservedStatus,
                  excludes,
                  excludeIndex,
                  excludePrefixLength,
                  exludeRangeBegin,
                  excludeRangeEnd,
                  excludeStatus,
                  bindings,
                  bindingsIndex,
                  bindingAddressPrefix,
                  bindingPreferLifeTime,
                  bindingValidLifeTime,
                  bindingNameServers,
                  bindingDomainName,
                  bindingDirAgent,
                  bindingDirAgentScopeList,
                  bindingServiceScope,
                  bindingNtpServers,
                  bindingNisDomainName,
                  bindingNisServers,
                  bindingNisPlusDomainName,
                  bindingNisPlusServers,
                  bindingKeepAliveInternal,
                  bindingStatus,
                  bindingActivitie,
                  statSolicit,
                  statAdvertise,
                  statRequest,
                  statRelease,
                  statReconfigure,
                  statReplyUnsError,
                  statReplyAuthError,
                  statReplyFormedError,
                  statReplyResourcesError,
                  statReplyClientRecError,
                  statReplyInvalidClientIpError,
                  statCharSetError

            }
      STATUS      current
```

```
      DESCRIPTION " "
  ::= { dhcpv6Groups 1 }

dhcpv6NotificationGroup NOTIFICATION-GROUP
      NOTIFICATIONS {
                        dhcpv6StateChange,
                        dhcpv6AddReservedAct,
                        dhcpv6DelReserved,
                        dhcpv6AddExclude,
                        dhcpv6ReservedStatus,
                        dhcpv6ExcludeStatus,
                        dhcpv6BindingActivitieChange,
                        dhcpv6StatReplyAuthError
                     }
      STATUS      current
      DESCRIPTION " "
  ::= { dhcpv6Groups 2 }
END
```

BIBLIOGRAFIA

[ALB97] ALBITZ, Paul; LIU, Cricket. *DNS and BIND*, 2nd edition. California: O'Reilly & Associates, Inc. 1997.

[ALE97] ALEXANDER, S.; DROMS; R . *IP DHCP Otions and BOOTP* Vendor Extensions. Request for Comments: 2132, March,1997.

[ATK95a] ATKINSON, R. *IP Authentication Header*. Request for Comments: 1826, August, 1995.

[ATK95b] ATKINSON, R. *IP Encapsulating Security Payload (ESP)*. Request for Comments: 1827, August, 1995.

[BLA94] BLACK, Uyless. *Network Management Standards:* SNMP, CMIP, TMN, MIBs and Object Libraries, 2nd edition. New York: McGraw-Hill, Inc. 1994.

[BLU97] BLUMENTHAL, Uri; WIJNEN, Bert. *Security Features of SNMPv3.*Simple Times: The Quartely NewsLetter of SNMP Technology, Comment, and Events, Volume 5, Number 1, December, 1997. http://www.simple-times.org/pub/simple-times/issues/5-1.html.

[BOR98] BORLAND, John. *How To Break The Net Before Breakfast*. TechWeb News, February, 1998.

[BOU98] BOUND, J. *Dynamic Host Configuration Protocol for IPv6*. Internet-Draft: draft-ietf-dhc-dhcpv6-12.txt, March, 1998.

[BRA93] BRADNER, Scott; MANKIN, Allison. *IP: Next Generation (IPng) White Paper*. Request for Comments: 1550, December, 1993.

[BRA95] BRADNER, Scott; MANKIN, Allison. *The Recommendation for IP Next Generation Protocol*. Request for Comments: 1752, January, 1995.

[BRA96] BRADNER, Scott; MANKIN, Allison. *IPng. Internet Protocol Next Generation*, second printing. Addinson-Wesley Publishing Company, Inc, 1996.

[BRI97] BRISA. *Arquitetura de Redes de Computadores. OSI e TCP/IP*, 2ª Edição. São Paulo: Makron Books, 1997.

[CAL92] CALLON, Ross. *TCP and UDP with Bigger Addresses (TUBA), A Simple Proposal for Internet Addressing and Routing.* Request for Comments: 1347, June 1992.

[CAS96a] CASE, J; MCCLOGHRIE, K.; ROSE, M.; WALDBUSSER, S. *Introduction to Community-based SNMPv2.* Request for Comments: 1901, January, 1996.

[CAS96b] CASE, J; MCCLOGHRIE, K.; ROSE, M.; WALDBUSSER, S. *Structure of Management for Version 2 of the Simple Network Management Protocol (SNMPv2).* Request for Comments: 1902, January, 1996.

[CAS96c] CASE, J; MCCLOGHRIE, K.; ROSE, M.; WALDBUSSER, S. *Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2).* Request for Comments: 1903, January, 1996.

[CAS96d] CASE, J; MCCLOGHRIE, K.; ROSE, M.; WALDBUSSER, S. *Conformance Statements for Version 2 of the Simple Network Management Protocol (SNMPv2).* Request for Comments: 1904, January, 1996.

[CAS96e] CASE, J; MCCLOGHRIE, K.; ROSE, M.; WALDBUSSER, S. *Protocolo Operations for Version 2 of the Simple Network Management Protocol (SNMPv2).* Request for Comments: 1905, January, 1996.

[CAS96f] CASE, J; MCCLOGHRIE, K.; ROSE, M.; WALDBUSSER, S. *Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2).* Request for Comments: 1906, January, 1996.

[CAS96g] CASE, J; MCCLOGHRIE, K.; ROSE, M.; WALDBUSSER, S. *Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2).* Request for Comments: 1907, January, 1996.

[CAS96h] CASE, J; MCCLOGHRIE, K.; ROSE, M.; WALDBUSSER, S. *Coexistence between Version 1 and Version 2 of the Internet-standard Network Management Framework.* Request for Comments: 1908, January, 1996.

[CER88] CERF, V. *IAB Recommendation for the Development of Internet Network Management Standards*. Request for Comments: 1052, April, 1988.

[CLA92] CLAFFY, Kimberly C.; POLYZOS, George C.; BRAUN, Hans-Werner. *Traffic Characteristics of T1 NSFNET Backbone*. California, 1992. Department of Computer Science and Engineering, University of California; Applied Network Research Group, San Diego Supercomputer Center.

[COH96] COHEN, Frederick. *Internet Holes. Packet Fragmentation Attacks*. 1996.

[COM92a] COMER, D.E.; STEVENS, D. L. *Internetworking with TCP/IP*, v.1. New Jersey: Prentice Hall, 1992.

[COM92b] COMER, D.E.; STEVENS, D. L. *Internetworking with TCP/IP*, v.2. New Jersey: Prentice Hall, 1992.

[COM92c] COMER, D.E.; STEVENS, D. L. *Internetworking with TCP/IP*, v.2. New Jersey: Prentice Hall, 1992.

[CON97] CONTA, A; DEERING, Steve.*Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6)*. Internet-draft: draft-ietf-ipngwg-icmp-v2-00.txt, October, 1997.

[CRA98] CRAWFORD, Matt; HINDEN, Robert. *Router Renumbering for IPv6*. Internet-draft:draft-ietf-ipngwg-router-renum-03.txt, March, 1998.

[CRO85] CROFT, B.; GILMORE, J. *Bootstrap Protocol*. Request for Comments: 951, September, 1985.

[DAN98a] DANIELE ,M. *IPv6 Management Information Base for the Transmission Control Protocol*. Internet-draft: draft-ietf-ipngwg-ipv6-tcp-mib-01.txt , January 1998.

[DAN98b] DANIELE ,M. *Management Information Base for the User Datagram Protocol* , Internet-draft: draft-ietf-ipngwg-ipv6-udp-mib-01.txt , January 1998.

[DEE95] DEERING, Steve; HINDEN, Robert. *Internet Protocol, Version 6 (IPv6) Specification*. Request for Comments: 1883, December, 1995.

[DEE97] DEERING, Steve; HIDEN, Robert. *Internet Protocol, Version 6 (IPv6) Specification*. Internet-draft:draft-ietf-ipngwg-ipv6-spec-v2-01.txt, November, 1997.

[DEF81a] DEFENSE ADVANCED RESEARCH PROJETCS AGENCY. *Internet Protocol.* Request for Comments: 791, September, 1981.

[DEF81b] DEFENSE ADVANCED RESEARCH PROJETCS AGENCY. *Transmission Control Protocol.* Request for Comments: 793, September, 1981.

[DRO97] DROMS; R . *IP Dynamic Host Configuration Protocol.* Request for Comments: 2132, March,1997.

[DUT97] DUTCHER, William. IP Addressing: Playing the Number. IP addresses are in short supply --- but there are ways to ease the crunch. *DataCommunications on web.* March, 1997. http://www.data.com/.

[FEI97] FEIT, Sidnie. *TCP/IP:Architecture, protocols and Implementation with IPv6 and IP Security.* Second edition. New York: McGraww-Hill,Inc. 1997.

[FRA94a] FRANCIS, Paul. *Pip Near-Term Architecture.* Request for Comments: 1621, May, 1994.

[FRA94b] FRANCIS, Paul. *Pip Header Processing.* Request for Comments: 1622, May, 1994.

[FIN84] FINLAYSON, R.; MANN, T.; MOGUL, J.; THEIMER, M. *A Reverse Address Resolution Protocol.* Request for Comments: 903, June, 1984.

[GAR97] GAREISS, Robin. *Is the Internet in Trouble?.* Data Communications on the Web, September, 1997.

[GIL96] GILLIAN, R; NORDMARK, E. *Transition Mechanisms for IPv6 Hosts and Routers.* Request for Comments: 1933, April, 1996.

[GRA96] GRAY, Matthew. *Internet Statistics. Growth and Usage of the Web and the Internet.* 1996. http://www.mit.edu/people/mkgray/net/.

[HAR97] HARRIGTON, David. *The Evolution of Architectural Cocepts in SNMPv3.*Simple Times: The Quartely NewsLetter of SNMP Technology, Comment, and Events, Volume 5, Number 1, December, 1997. http://www.simple-times.org/pub/simple-times/issues/5-1.html.

[HAS98a] HASKIN, Dimitry; ONISHI, Steve. *Management Information Base for IP Version 6: Textual Conventions and General Group.* Internet-draft: draft-ietf-ipngwg-ipv6-mib-04.txt , February 1998.

[HAS98b] HASKIN, Dimitry; ONISHI, Steve . *Management Information Base for IP Version 6: ICMPv6 Group.* Internet-draft : draft-ietf-ipngwg-ipv6-icmp-mib-02.txt , January 1998.

[HIN94] HINDEN, Robert. *Simple Internet Protocol Plus White Paper.* Request for Comments:1710, October, 1994.

[HIN95a] HINDEN, Robert. *IP Next Generation Overview.* May, 1995. http://playground.sun.com/pub/INET-IPng-Paper.html#CH8.

[HIN95b] HINDEN, Robert; DEERING, Steve. *IP Version 6 Addressind Architecture.* Request for Comments: 1884, December, 1995

[HIN97a] HINDEN, Robert; DEERING, Steve. *IP Version 6 Multicast Address Assignments.* Internet-draft: draft-ietf-ipngwg-addr-multicast-assgn-04.txt, July, 1997.

[HIN97b] HINDEN, Robert; DEERING, Steve. *IP Version 6 Specification.* Internet-draft: draft-ietf-ipngwg-ipng-spec-v2-01.txt, November, 1997.

[HIN97c] HINDEN, Robert; DEERING, Steve. *IP Version 6 Testing Address Allocation.* Internet-draft: draft-ietf-ipngwg-testv2-addralloc-01.txt, July, 1997.

[HIN97d] HINDEN, Robert; DEERING, Steve. *TLA and NLA Assigment Rules.* Internet-draft: draft-ietf-ipngwg-tla-assignment-02.txt, November, 1997.

[HIN98a] HINDEN, Robert; DEERING, Steve. *IP Version 6 Addressind Architecture.* Internet-draft: draft-ietf-ipngwg-addr-arch-v2-06.txt, January, 1998.

[HIN98b] HINDEN, Robert; O'DELL, M. *An IPv6 Aggregatable Global Unicast Address Format.* Internet-draft: draft-ietf-ipngwg-unicast-aggr-04.txt, March, 1998.

[HUI95] HUITEMA, Christian. *Routing in the Internet.* New Jersey: Prentice Hall, 1995.

[HUI97] HUITEMA, Christian. *IPv6. The New Internet Protocol,* second edition.New Jersey: Prentice Hall, 1997.

[IBM95] IBM. *TCP/IP Tutorial and Technical Overview*. Fifth Edition: : International Business Machines Corporation, June,1995. http://www.rs6000.ibm.com/resource/aix_resource/Pubs/redbooks/htmlbooks/gg243376.04/3376fm.html.

[IBM97] IBM. *TCP/IP AIX Version 4.3 System Management Guide: Communications and Networks*. Second Edition: International Business Machines Corporation,October,1997. http://www.austin.ibm.com/doc_link/en_US/a_doc_lib/aixbman/commadmn/toc.htm.

[IEE97] IEEE. *Guidelines for 64-bit Global Identifier (EUI-64) Registration* Authority, May, 1997. http://standards.ieee.org/regauth/oui/tutorials/EUI64.html.

[IET98] IETF. *IP Next Generation (IPng)*. Internet Engineering Task Force, 1998. http://www.ietf.cnri.reston.va.us/html.charters/ipngwg-charter.html.

[IPM98] IPMA. Performance and Analysis Project. *Selected Papers on Internet Statistics and Routing Stability*. University of Michigan; Merit Network, 1998. http://www.merit.edu/ipma/docs/paper.html.

[IPM98a] IPMA- Performance and Analysis Project. *Network Internet Performance and Statistics in the Press* .University of Michigan; Merit Network, 1998. http://www.merit.edu/ipma/docs/paper.html.

[JOH98] JOHNSON, David B.; PERKINS, Charles. *Mobility Support in IPv6*. Internet-draft: draft-ietf-mobileip-ipv6-05.txt, March, 1998.

[LAB97] LABOVITZ, Craig; Malan, G. Robert; Jahanian, Farnam. *Internet Routing Instability*. Michigan, 1997. Department of Electrical Engineering and Computer Science, University of Michigan.

[LOT92] LOTTOR, M. *Internet Growth (1981-1991)*. Request for Comments: 1296, January, 1992.

[MCC96a] MCCANN, J.; DEERING, Steve; MOGUL, J. *Path MTU Discovery*. Request for Comments.1981, August, 1996.

[MCC96b] MCCLOGHRIE, K; ROSE, Marshal. *Management Information Base for Network Management of TCP/IP-based internets: MIB-II*. Request for Comments.1213, March, 1991.

[MCG94] MCGOVERN, M.; ULMANN, Robert. *CATNIP: Common Architecture for the Internet*. Request for Comments: 1707, October, 1994.

[MET96] METCALFE, Robert. From the Ether. The number show how slowly the Internet runs today. *INFOWORLD Electric*. September, 1996. http://www.infoworld.com/.

[MOG90] MOGUL, J.; DEERING, Steve. *Path MTU Discovery*. Request for Comments: 1191, November, 1990.

[MUN98] MUNDY, Russ. *SNMP Version 3 (snmpv3)*. April, 1998. http://www.ietf.org/html.charters/snmpv3-charter.html.

[NAR98] NARTEN, Thomas; NORDMARK, Erik. *Neighbor Discovery for IP Version 6 (IPv6)*. Internet-draft: draft-ietf-ipngwg-discovery-v2-02.txt, February, 1998.

[PRA95] PRAS, Aiko; *Network Management Architectures*. Enschede, 1995. Thesis University of Twente.

[PER98] PERKINS, C. *Extensions for the Dynamic Host Configuration Protocol for IPv6*. Internet-draft: draft-ietf-dhc-v6exts-09.txt, March, 1998.

[POS80] POSTEL, J. *User Datagram Protocol*. Request for Comments: 768, August, 1980.

[POS81] POSTEL, J.. *Internet Control Message Protocol*. Request for Comments: 792, September, 1981.

[PIS93a] PISCITELLO, D. *Assignment of System Identifiers for TUBA/CLNP Hosts*. Request for Comments: 1526, September, 1993.

[PIS93b] PISCITELLO, D. *Use of ISO CLNP in TUBA Environments*. Request for Comments: 1561, December, 1993.

[PLU82] PLUMMER, David C. *An Ethernet Address Resolution Protocol*. Request for Comments: 826, November, 1982.

[ROS94] ROSE, Marshall T. *The Simple Book: An Introduction to Internet Managment*. Second Edition. New Jersey: Prentice Hall. 1994.

[RUT97] RUTKOWSKI, Tony. *Internet Trends*. February, 1997. http://www.genmagic.com/Internet/Trends/.

[SCO98] SCOTT, Mace. The Next Internet. Layer 3 routing, IPv6, and IP Multicast are all technologies set to take off – or stay on hold. *BYTE*. January, 1998. http://www.byte.com/.

[SIY92] SIYAN, K. *An IP Address Extension Proposal*. Request for Comments: 1365, September, 1992.

[SIM94] SIMPSON, W. *IPng Mobility Considerations*. Request for Comments: 1698, August, 1994.

[SOA95] SOARES, Luiz Fernando Gomes; LEMOS, Guido; COLCHER, Sérgio. *Redes de Computadores. Das LANs MANs e WANs às Redes ATM*. Rio de Janeiro: Editora Campus, 1995.

[STA96a] STALLINGS, William. The New and Improved Internet Protocol. Enhancements to the Internet Protocol let it support more address and handle Web functions. *BYTE*. September, 1996. http://www.byte.com/.

[STA96b] STALLINGS, William. Internet Armor. Secure IP fule the drive for secure transactions and communications over the Internet. *BYTE*. December, 1996. http://www.byte.com/.

[STE97] STERN, Morgan. Extend Your Net with VPNs. These virtual private network packages let you use the Internet as your own private WAN. *BYTE*. November, 1997. Institute of Electrical and Electronics Engineers.

[TAN97] TANENBAUM, Andrew. S. *Redes de Computadores*, tradução da terceira edição. Rio de Janeiro: Editora Campus, 1997.

[THO96] THOMPSON, Susan. *IPv6 Stateless Address Autoconfiguraton*. Request for Comments: 1971, August ,1996.

[THO98a] THOMPSON, Susan. HUITEMA, Christian. *DNS Extensions to support IP version 6*. Internet draft: draft-ietf-ipngwg-aaaa-03.txt, February, 1998.

[THO98b] THOMPSON, Susan. *IPv6 Stateless Address Autoconfiguraton*. Internet-draft: draft-ietf- ipngwg-addrconf-v2-02.txt, February, 1998

[ULL93] ULLMANN, Robert. *TP/IX: The Next Internet*. Request for Comments: 1475, June, 1993.

[VEI97] VEIZADES, J.; GUTTMAN, E.; PERKINS, C.; KAPLAN, S. *Service Location Protocol*. Request for Comments: 2165, June, 1997.

[WIZ98] Wizards, Network. Internet Domain Survey. February, 1998. http://www.nw.com/zone/WWW/top.html02.txt.